

SOLVE:EPS

User's Guide
1st Edition



Manual History

Edition	Publication Number	Product Version	Min. MS Level	Publish Date
1st Edition		Version 5.2		May, 1999

New editions incorporate any updates issued since the previous edition.

SOLVE:EPS™ User's Guide, 1st Edition

This document was prepared to assist licensed users with the use of SOLVE:EPS™; its contents may not be used for any other purpose without prior written permission, and this manual is subject to the terms and conditions of the applicable license agreement. The material contained herein is supplied without representation or warranty of any kind. Sterling Software and its subsidiaries, therefore, assume no responsibility and shall have no liability of any kind arising from the supply or use of this document or the material contained herein.

References in this manual to Sterling Software products, programs, or services do not imply that Sterling Software intends to make these available in all countries in which Sterling Software operates.

All rights reserved, including the right to reproduce this document or any portion thereof in any form.

Copyright© 1999, Sterling Software, Inc.

The status symbols ® and ™, as used to identify Sterling Software trademarks herein, refer to the status of Sterling Software trademarks as pending or registered in the U.S. Patent and Trademark Office as of July 1992. Sterling Software has applied for and has been granted registrations for their trademarks in various countries throughout the world. Sterling Software will act to protect its trademark rights worldwide.

INFO/MASTER is a registered trademark of Sterling Software, Inc. in the U.S.A. and/or other countries. AOM, EASINET, External Interface Package, EIP, Expert System Foundation, ESF, Inter-Management Services Connection/Extended Function, INMC/EF, Multiple Application Interface, MAI, MAI/EF, Network Control Language, NCL, Network Control System, NCS, Network Error Warning System, NEWS, Network Tracking System, NTS, NET/MAIL, SOLVE, SOLVE:Access, SOLVE:Asset, SOLVE:Attach, SOLVE:Automation, SOLVE:Central, SOLVE:Change, SOLVE:Commander, SOLVE:Configuration, SOLVE:Connect, SOLVE:CPT, SOLVE:EPS, SOLVE:Information, SOLVE:Link, SOLVE:Monitor, SOLVE:Netmaster, SOLVE:Netmaster for TCP/IP, SOLVE:Operations, SOLVE:Problem, SOLVE:Stat, and SOLVE:TCPaccess are trademarks of Sterling Software, Inc. and or its subsidiaries.

SOLVE:EPS User's Guide

Table of Contents

Chapter 1 SOLVE:EPS Operator Commands

Overview	1 - 2
Abbreviations	1 - 2
Input Sources	1 - 2
Command Syntax	1 - 3
Dynamic Parameters	1 - 3
Models and Defaults	1 - 4
Message and Log Controls	1 - 4
SOLVE:EPS Commands	1 - 5
CLEAR	1 - 5
DEFINE ALTERNATECONSOLE	1 - 6
DEFINE ALTERNATELOG	1 - 6
DEFINE CONVERTER	1 - 6
DEFINE INPROCESSOR	1 - 7
DEFINE OUTPROCESSOR	1 - 8
DEFINE QUEUE	1 - 9
EXECUTE	1 - 10
HOLD	1 - 11
PROMPT	1 - 11
PURGE	1 - 13
REFRESH	1 - 14
RELEASE	1 - 15
RESET	1 - 16
ROUTE	1 - 17
SET ALTERNATECONSOLE	1 - 18
SET ALTERNATECONSOLE Parameters	1 - 18
SET ALTERNATELOG	1 - 23
SET ALTERNATELOG Parameters	1 - 23
SET CONVERTER	1 - 27
SET SOLVE:EPS	1 - 28
SET EPS Parameters	1 - 28
SET INPROCESSOR	1 - 31

SET INPROCESSOR Parameters	1 - 31
Job Defaults	1 - 36
BSD-Specific Parameters	1 - 40
NJE-Specific Parameters	1 - 41
SSI-Specific Parameters	1 - 45
VPRT-Specific Parameters	1 - 46
SET JOB	1 - 48
SET JOB Parameters	1 - 48
Job Defaults	1 - 49
SET LOG	1 - 52
SET LOG Parameters	1 - 52
SET OUTPROCESSOR	1 - 56
SET OUTPROCESSOR Parameters	1 - 56
BSD-Specific Parameters	1 - 60
DSO-Specific Parameters	1 - 60
NJE-Specific Parameters	1 - 61
SET QUEUE	1 - 62
SET QUEUE Parameters	1 - 62
BSD-Specific Parameters	1 - 70
DSO-Specific Parameters	1 - 78
SET SPOOL	1 - 81
SET SPOOL Parameters	1 - 81
SHOW ALTERNATECONSOLE	1 - 86
SHOW ALTERNATELOG	1 - 88
SHOW CONVERTER	1 - 89
SHOW EPS	1 - 90
SHOW INPROCESSOR	1 - 93
SHOW JOB	1 - 96
SHOW LOG	1 - 99
SHOW OUTPROCESSOR	1 - 101
SHOW QUEUE	1 - 103
SHOW SPOOL	1 - 106
SHUTDOWN	1 - 109
SPIN	1 - 109
START	1 - 110
STOP	1 - 111

Chapter 2 **SOLVE:EPS Operation**

SOLVE:EPS Execution	2 - 2
SOLVE:EPS Job Control Language (JCL) Statements	2 - 2

Customizing the JCL	2 - 3
Changes to the SOLVE:EPS Checkpoint File	2 - 3
Checkpoint File Conversion Utility	2 - 3
Changes From Checkpoint File Level 1 to Checkpoint File Level 2	2 - 4
Running the Checkpoint File Conversion Utility	2 - 4
JCL Explanation	2 - 7
Execution Options	2 - 7
Starting and Stopping SOLVE:EPS	2 - 8
Starting SOLVE:EPS	2 - 8
Stopping SOLVE:EPS	2 - 8
Recovery Action After Unexpected Termination	2 - 8
SOLVE:EPS Commands Usage	2 - 9
Status Display	2 - 9
Trace Control	2 - 9
Dynamic Definition	2 - 11
Stopping and Restarting Virtual Printers	2 - 11
Using SOLVE:EPS-Administration	2 - 11
Message and Log Controls	2 - 12
Error Messages	2 - 12
Informational Messages	2 - 12
Response Messages	2 - 12
Trace Messages	2 - 13

Chapter 3 Using SOLVE:EPS

Printing on Remote Printers	3 - 2
Using the JES2/NJE Interface	3 - 2
PRINTDS	3 - 3
Using the Virtual Printer Emulator	3 - 3
Using LU3	3 - 4
LU3 Limitations	3 - 4
Printing on JES Printers	3 - 5
Using the UNIX lpr Command	3 - 5
The -P Option	3 - 6
The -C Option	3 - 7
Supported LPD Command Options	3 - 8
Example of lpr Usage	3 - 10

Appendix A User Exits

Overview	A - 2
----------------	-------

User Exit 1 – NJE Header/Trailer	A - 3
User Exit 2 - Control File Line	A - 4
Dispatch User Exit	A - 6
Specifying the Dispatch User Exit	A - 6
Data Objects Available to the Dispatch User Exit	A - 7
queue Object Fields	A - 8
job Object Fields	A - 9
boolean Object Fields	A - 10
rc Object Fields	A - 11
return Object Fields	A - 11
message Object Fields	A - 12
Sample Dispatch User Exit	A - 12
Tips	A - 12
Changing Control Strings	A - 12
Requeuing a Job	A - 13
Holding a Job	A - 13
Changing a Job's Dispatching Priority	A - 14
Dataset User Exit	A - 15
Specifying the Dataset User Exit	A - 15
Assembler Dataset User Exits	A - 15
EPSUSEX Dataset User Exits	A - 16
Assembler Interface	A - 17
SOLVE:EPS Dataset User Exit Assembler Interface	A - 17
EPSUSEX Interface	A - 19
Basic Processing	A - 19
Data Objects Available to the Dataset User Exit	A - 20
Calling the Dataset User Exit	A - 28
Sample Dataset User Exits	A - 28
Tips	A - 28
Setting Multiple returnstring Options	A - 28
Inserting Block Letters	A - 29
Determining the Beginning and the End of a SOLVE:EPS Job	A - 29
Querying the Disposition Field	A - 29
Querying Data Type and Subtype Fields	A - 30
Querying the nobanner Field	A - 30
Including Multiple Pieces of Information in returnstring Text	A - 30
Determining When to Insert What Data	A - 31
Leading Zeros	A - 31
CONVERTSELECTOR Outprocessor User Exit	A - 32

rc Object Fields for CONVERTSELECTOR Outprocessor User Exit	A - 33
return Object Fields for CONVERTSELECTOR Outprocessor User Exit	A - 33
Initialization User Exits	A - 34
LU1 Converter	A - 34
Most Useful LU1 Converter Initialization User Exit Options	A - 34
Objects Available to the LU1 Converter Initialization User Exit	A - 35
Translate/Only Converter Initialization User Exit	A - 40
Binary Converter Initialization User Exit	A - 41
Objects Available to the Translate/Only and Binary Converter Initialization User Exit	A - 42
JES-to-ASCII Converter	A - 43
Objects Available to the JES-to-ASCII Converter Initialization User Exit	A - 44
Command Security User Exit	A - 47
Main Entry Point	A - 47
VerifyUser	A - 48
AuthorizeCommand	A - 48
ReleaseUser	A - 48
Terminate	A - 49
IsEachSignonUnique	A - 49
Sample SAF Based Command Security Exit T07XXCS2	A - 49
Implementing Command Security Using RACF	A - 50
Response User Exit	A - 53
Shared Fields Tables	A - 54
language Object Fields	A - 54
converter Object Fields	A - 54
rc Object Fields for Binary, JES-to-ASCII, LU1, and Translate/Only Initialization User Exits	A - 55
return Object Fields for Binary, JES-to-ASCII, LU1, and Translate/Only Initialization User Exits	A - 55
translatedirection Object Fields	A - 56
dbcssupport Object Fields	A - 56
formfeed Object Fields	A - 57
Overriding Character Handling for LU1	A - 58
Using the Default LU1 Converter	A - 58
Defining a New LU1 Converter	A - 59
Including Escape Sequences in LU1 Data	A - 60
Ignoring End-of-Message (EM) Characters	A - 60
Applying Character Handling Selectively	A - 61

Appendix B EPSUSEX User Exit Language

Overview	B - 2
----------------	-------

Data Types	B - 2
Strings	B - 2
Numbers	B - 2
Bits	B - 2
Data Categories	B - 3
Objects with fields	B - 3
Literals	B - 3
String Literals	B - 3
Numeric Literals	B - 3
Hexadecimal String literals	B - 4
Bit Literals	B - 4
Index Object References	B - 5
Statement Types	B - 6
Conditionals	B - 6
if, elseif, and else Statements	B - 6
Logical Expressions	B - 7
String Comparators	B - 7
Numeric Comparators	B - 7
Bit Comparators	B - 8
Assignments	B - 8
String Assignment Operators	B - 8
Number Assignment Operators	B - 8
Bit Assignment Operators	B - 9
Assignment of Numbers to Strings	B - 9
break Statement	B - 9
Comments	B - 9
EPSUSEX Keywords	B - 10
Implemented Exits	B - 10

Appendix C Using SOLVE:EPS-Administration

Overview	C - 3
Commands	C - 4
EXECUTE	C - 4
NEWQUEUE	C - 4
OPTIONS	C - 4
RELOAD	C - 4
REFRESH	C - 4
SHUTDOWN	C - 4
SPIN	C - 5

Parameters	C - 6
TCP Address Space Name	C - 6
Remote Machine Name	C - 6
Remote Port	C - 6
Remote User ID	C - 6
Remote Password	C - 6
Inactivity Timeout	C - 6
Debug	C - 7
Native Interface Modeling	C - 7
Queues and Outprocessors	C - 8
Queue Filter	C - 8
Dummy Outprocessor	C - 8
Zoom Outprocessor	C - 8
Zoom Queue	C - 9
New Queue	C - 9
Route Queue	C - 9
Jobs	C - 10
Job Filters	C - 10
Reset Job	C - 10
Zoom Job	C - 10
Inprocessors	C - 11
Zoom Inprocessor	C - 11
Spool	C - 11
Logs	C - 11
SOLVE:EPS	C - 12
SOLVE:EPS Command Shell	C - 12
Frequently Asked Questions	C - 13
Error Messages	C - 13
Other Communication Failures	C - 13
Usage Notes	C - 14

Index

1

SOLVE:EPS Operator Commands

This chapter describes the SOLVE:EPS (Enterprise Print Services) commands. It includes these sections:

- [“Overview” on page 1-2](#)
Provides a brief description of how SOLVE:EPS and other systems use the configuration statements.
- [“SOLVE:EPS Commands” on page 1-5](#)
Describes the SOLVE:EPS commands.

Overview

The SOLVE:EPS commands are the basis for defining and controlling the **EPS** environment. There is not a separate syntax for defining the environment at startup time versus manipulating the environment during execution. This provides a consistent way to define and control SOLVE:EPS.

The configuration of SOLVE:EPS Version 5.2 is very different from the configuration of EPS Version 2.0. The EPS Version 2.0 configuration file was replaced with **DEFINE** and **SET** commands in Version 4.1. Parameters of some of these commands correspond to the “configuration statements” of EPS Version 2.0. The appendix “Comparative Command Tables” in the ***SOLVE:EPS Installation and Administration Guide*** provides a comparison of the EPS Version 2.0 configuration statements and the EPS Version 4.1 commands. These Version 4.1 commands are the same in Version 5.2 of SOLVE:EPS.

Abbreviations

Command verbs and parameters can be abbreviated to the smallest number of characters that make them unique from the rest of the verbs or keywords; this includes any aliases that parameters or verbs might have.

For example, the **CLEAR** command could be abbreviated as C, CL, CLE or CLEA. The **DEFINE** command could be abbreviated as DEF, DEFI, or DEFIN. However, the **RESET** command must use at least the first three letters, RES, in order to be unique from **REFRESH** and **RELEASE**.

This example uses an abbreviation of I for INPROCESSOR in the **SET INPROCESSOR** command:

```
SET I inp_name NOQUEUEDEFAULT
```

Input Sources

Input for the SOLVE:EPS commands can be from any of these three sources:

- The console, via the MVS **MODIFY** command interface
- The console, in reply to an outstanding WTOR that was requested via the **PROMPT** command
- A member of a PDS pointed to by the DDNAME PARMLIB

The member name can be a parameter passed on the execute statement of the startup JCL when SOLVE:EPS is started, or it can be specified on the SOLVE:EPS **EXECUTE** command.

Note: When entering commands from source that only supports uppercase characters (for example, an MVS console) you must use a LOWERCASEINDICATOR character to get the SOLVE:EPS command processor to lowercase. Read [“LOWERCASEINDICATOR/*NOLOWERCASEINDICATOR” on page 1-29](#) later in this chapter.

Command Syntax

Typically, an SOLVE:EPS command consists of a verb, an object type, and an object name, along with some parameters. Each word of the command is separated with one or more spaces. Continuation is provided for two of the three sources (read [“Input Sources” on page 1-2](#)). For the PROMPT source and EXECUTE source, continuation is signified by using a dash (-) as the last character of the input line. SOLVE:EPS recognizes the continuation character and processes the next input line as a continuation of the current input line.

This is a typical SOLVE:EPS command:

SET INPROCESSOR *obj_name* **QUEUEDEFAULT** *q_name*

Syntax Description

SET	The command verb.
INPROCESSOR	The object type.
<i>obj_name</i>	The name of the object to be set.
QUEUEDEFAULT	A command parameter.
<i>q_name</i>	The parameter value.

This command shows multiple commands setting the same object:

SET JOB j2 QUEUE *abc*
SET JOB j2 JOBNAME *xyz*
SET JOB j2 CLASS x FORMS *forms*

This command shows continuation:

SET JOB j2 QUEUE *abc* -
JOBNAME *xyz* -
CLASS X FORMS *forms*

Dynamic Parameters

Some parameters are specified as “dynamic,” which means they are settable in any state. A parameter that is “not dynamic” can be set only in certain states.

Models and Defaults

Models are special SOLVE:EPS objects. Their purpose is to serve as templates when defining other objects. Models are created with the **DEFINE** command by specifying the MODEL parameter and, optionally, the DEFAULT parameter. Once a model has been defined, it can be altered with the **SET** command, and referenced either implicitly or explicitly by subsequent **DEFINE** commands.

To explicitly use a model when defining an object, use the LIKE parameter of the **DEFINE** command. To implicitly use a model, designate a default model by specifying the DEFAULT parameter when defining or setting the model. Any parameters specified on the **DEFINE** command will override those inherited from the model.

When a model is implicitly or explicitly employed during object definition, the object immediately inherits the attributes of the model. Subsequent changes to the model's attributes, including its default status, only impact objects defined thereon. Objects that had been defined prior to the changes made to the model are not affected.

The only SOLVE:EPS objects that support model definitions are Inprocessors, Outprocessors, and queues.

Message and Log Controls

SOLVE:EPS produces messages that will be written to the MVS operator console and logged to a SOLVE:EPS designated log. These are the types of messages there are:

- Error messages
- Informational messages
- Response messages
- Tracing messages

All messages except response messages are controllable via **SET** commands for each object. For more information, read the descriptions of the **SET** commands in this chapter.

Read the chapter "[SOLVE:EPS Operation](#)", for descriptions of the types of messages used by SOLVE:EPS. The **SOLVE:EPS Error Messages** manual describes all of the messages used by SOLVE:EPS.

SOLVE:EPS Commands

This section describes all of the SOLVE:EPS commands.

CLEAR

The **CLEAR** command removes defined objects. An object must be stopped before it can be cleared. Additionally, an Outprocessor cannot be cleared if it is referenced by any active (i.e., not stopped) queues.

The alias for the **CLEAR** command is ERASE.

CLEAR *obj_type obj_name* [**MODEL**]

Syntax Description

<i>obj_type</i>	The type of object to clear. Types of objects that can be cleared are ALTERNATECONSOLE, ALTERNATELOG, CONVERTER, INPROCESSOR, OUTPROCESSOR, and QUEUE. Aliases for ALTERNATECONSOLE are ACN or ALTCONS; aliases for ALTERNATELOG are ALG or ALTLOG.
<i>obj_name</i>	The name of the object to be cleared.
MODEL	Specifies that this object model be cleared. This parameter applies only to object types Inprocessor, Outprocessor, and Queue.

Note: You can use an asterisk (*) to clear all objects of the specified type.

CLEAR ALTERNATECONSOLE *cons_name*

CLEAR ALTERNATELOG *log_name*

CLEAR CONVERTER *conv_name*

CLEAR INPROCESSOR *abc*

CLEAR OUTPROCESSOR *xyz*

CLEAR QUEUE *jkl*

DEFINE ALTERNATECONSOLE

The **DEFINE ALTERNATECONSOLE** command defines a task that allows remote commands to be entered from a Telnet session or ISPF SOLVE:EPS-Administration (SOLVE:EPS-Admin) session.

The aliases for DEFINE ALTERNATECONSOLE is CREATE ALTERNATECONSOLE; aliases for ALTERNATECONSOLE are ACN or ALTCONS.

DEFINE ALTERNATECONSOLE *cons_name* **TYPE** **TCP** [*PARAMETERS*]

Syntax Description

<i>cons_name</i>	Name of the alternate console to be defined. <i>cons_name</i> can be up to thirty-two nonblank alphanumeric characters.
TYPE	Type of alternate console to be defined.
<i>PARAMETERS</i>	Any parameters described with the SET ALTERNATECONSOLE command.

DEFINE ALTERNATELOG

The **DEFINE ALTERNATELOG** command defines a task that will send log messages to the e-Control subsystem for automated log reporting of events.

The alias for DEFINE ALTERNATELOG is CREATE ALTERNATELOG; aliases for ALTERNATELOG are ALG or ALTLOG.

DEFINE ALTERNATELOG *log_name* **TYPE** **ECONTROL** [*PARAMETERS*]

Syntax Description

<i>log_name</i>	Name of the alternate log to be defined. <i>log_name</i> can be up to thirty-two nonblank alphanumeric characters.
TYPE	Type of alternate log to be defined.
<i>PARAMETERS</i>	Any parameters described with the SET ALTERNATELOG command.

DEFINE CONVERTER

The **DEFINE CONVERTER** command defines a new converter to SOLVE:EPS. It takes effect immediately.

DEFINE CONVERTER *conv_name* **INITMODULE** *mod_name* [**INITEXIT** *mem_name*]

Syntax Description

<i>conv_name</i>	Specifies the name of the converter, up to 32 non-blank characters.
<i>mod_name</i>	Specifies the name of the converter initialization module.
<i>mem_name</i>	Specifies the name of an optional converter Initialization User Exit written in EPSUSEX (described in the appendix “EPSUSEX User Exit Language”).

DEFINE INPROCESSOR

The **DEFINE INPROCESSOR** command creates an Inprocessor object. The Inprocessor object represents an input process within SOLVE:EPS.

The Inprocessor must have a name and a type; other parameters are optional.

Note: Parameters for the **DEFINE INPROCESSOR** command are described with the **SET INPROCESSOR** command.

The alias for **DEFINE INPROCESSOR** is **CREATE INPROCESSOR**.

DEFINE INPROCESSOR *inp_name* **TYPE** NJE | SSI | BSD | VPRT
[**MODEL** | **LIKE** *likename*] [**PARAMETERS**]

Syntax Description

<i>inp_name</i>	Name of the Inprocessor to be defined. <i>inp_name</i> can be up to thirty-two nonblank alphanumeric characters.
TYPE	Type of Inprocessor to be defined. This parameter can be specified only at the time the Inprocessor is defined. The TYPE parameter has no default; you must enter one of these types: NJE—Specifies that the Inprocessor receive jobs via an NJE link. SSI—Specifies that the Inprocessor receive jobs via the subsystem interface. BSD (or LPD)—Specifies that the Inprocessor receive jobs via a sockets interface using BSD LPR protocol (in other words, the Inprocessor of an LPD server). The alias for the BSD parameter of DEFINE INPROCESSOR is LPD. VPRT—Specifies that the Inprocessor receive jobs from VTAM applications that use LU1 or LU3 SNA printing protocols.
MODEL	Specifies that this Inprocessor entry is to be a model which can be referenced by other DEFINE INPROCESSOR commands by using the LIKE parameter.
LIKE <i>likename</i>	Specifies that this definition use the model Inprocessor entry specified by <i>likename</i> . Any parameters defined or set on the model Inprocessor will be set in this new Inprocessor definition. If a SET INPROCESSOR DEFAULT has been done for a previously defined Inprocessor model, a LIKE parameter need not be specified on subsequent DEFINE INPROCESSOR commands for the new definition to assume the parameters of the DEFAULT definition. NOTE: The LIKE parameter and MODEL parameter are mutually exclusive.
PARAMETERS	Any parameters described with the command “SET INPROCESSOR” on page 1-31 .

DEFINE OUTPROCESSOR

The **DEFINE OUTPROCESSOR** command creates an Outprocessor object. The Outprocessor object represents an output process within SOLVE:EPS.

The Outprocessor must have a name and a type; other parameters are optional.

Note: Parameters for the **DEFINE OUTPROCESSOR** command are described with the **SET OUTPROCESSOR** command.

The alias for **DEFINE OUTPROCESSOR** is **CREATE OUTPROCESSOR**.

DEFINE OUTPROCESSOR *outp_name* **TYPE** **NJE** | **BSD** | **DSO** | **JES**
[**MODEL** | **LIKE** *likename*] [*PARAMETERS*]

Syntax Description

<i>outp_name</i>	Name of the Outprocessor to be defined. <i>outp_name</i> can be up to thirty-two nonblank alphanumeric characters.
TYPE	Type of Outprocessor to be defined. This parameter can be specified only when the Outprocessor is defined. The TYPE parameter has no default; you must enter one of these types. • NJE—Specifies that the Outprocessor send jobs via an NJE link. • BSD (or LPR)—Specifies that the Outprocessor send jobs via a sockets interface using BSD LPR protocol (in other words, the Outprocessor is an LPD client). The alias for the BSD parameter of DEFINE OUTPROCESSOR is LPR. • JES—Specifies that the Outprocessor send jobs to the JES subsystem via dynamic allocation. • DSO—Specifies that the Outprocessor send jobs via a sockets interface directly to remote printers.
MODEL	Specifies that this Outprocessor entry be set up as a model which can be referenced by other DEFINE OUTPROCESSOR commands by using the LIKE parameter.
LIKE <i>likename</i>	Specifies that this definition will use the model Outprocessor entry specified by <i>likename</i> . Any parameters defined or set on the model Outprocessor will be set in this new Outprocessor definition. If a SET OUTPROCESSOR DEFAULT has been done for a previously defined Outprocessor model, a LIKE parameter need not be specified on subsequent DEFINE OUTPROCESSOR commands for the new definition to assume the same parameters as the DEFAULT definition.
<i>PARAMETERS</i>	Any parameters described with the command “SET OUTPROCESSOR” on page 1-56 .

Note: The **LIKE** parameter and **MODEL** parameter are mutually exclusive.

DEFINE QUEUE

The **DEFINE QUEUE** command creates a queue object. A queue represents a destination within SOLVE:EPS for jobs.

The queue must have at least a name; other parameters are optional. However, an Outprocessor must be specified in order to start the queue.

Note: Parameters for the **DEFINE QUEUE** command are described with the **SET QUEUE** command.

The alias for **DEFINE QUEUE** is CREATE QUEUE.

DEFINE QUEUE *q_name* [**MODEL** | **LIKE** *likename*] [*PARAMETERS*]

Syntax Description

<i>q_name</i>	Name of the queue to be defined. <i>q_name</i> can be up to thirty-two nonblank alphanumeric characters.
MODEL	Specifies that this queue entry is to be set up as a model which can be referenced by other DEFINE QUEUE commands by using the LIKE parameter.
LIKE <i>likename</i>	Specifies that this definition will use the model queue entry specified by <i>likename</i> . Any parameters defined or set on the model queue will be set in this new queue definition. If a SET QUEUE DEFAULT has been done for a previously defined queue model, a LIKE parameter need not be specified on subsequent DEFINE QUEUE commands for the new definition to assume the same parameters as the DEFAULT definition.
<i>PARAMETERS</i>	Any parameters described with the command “SET QUEUE” on page 1-62 .

Note: The LIKE parameter and MODEL parameter are mutually exclusive.

EXECUTE

The **EXECUTE** command tells SOLVE:EPS to process commands from a member of the PDS referred to by the PARMLIB DD statement in the SOLVE:EPS startup JCL. The command processor reads the PDS member and processes all commands in the member.

Aliases for **EXECUTE** are DO, INCLUDE, and PERFORM.

Note: The **EXECUTE** command itself can be in the PDS member but be careful that you do not cause loops with this facility. Otherwise, unpredictable results will occur.

EXECUTE *mem_name*

Syntax Description

<i>mem_name</i>	Name of the PDS member where commands to be processed reside.
-----------------	---

HOLD

The **HOLD** command prevents jobs from being processed by an Outprocessor.

Only ready jobs can be held.

```
HOLD eps_job_id1 [ eps_job_id2 ... | THRU eps_job_idn ]
    [ CLASS class ] [ DESTINATION dest ] [ FORMS form ] [ ORPHANS ]
    [ QUEUE q_name ] [ USER user ]
```

Syntax Description

<i>eps_job_idn</i>	ID of the SOLVE:EPS jobs to be held. NOTE: You can use the asterisk (*) to hold all jobs in SOLVE:EPS.
THRU	Specifies the range of jobs to be held. Range is 1 to 99999.
CLASS <i>class</i>	Specifies a class to restrict the range of jobs. There can be multiple CLASS job qualifiers on the command.
DESTINATION <i>dest</i>	Specifies a destination to restrict the range of jobs. There can be multiple DESTINATION job qualifiers on the command.
FORMS <i>form</i>	Specifies a form to restrict the range of jobs. There can be multiple FORMS job qualifiers on the command.
ORPHANS	Specifies a restriction of the range of jobs to those on the orphan queue.
QUEUE <i>q_name</i>	Specifies an SOLVE:EPS queue to restrict the range of jobs; if QUEUE is not specified, all queues will be held. There can be multiple QUEUE job qualifiers on the command.
USER <i>user</i>	Specifies a user to restrict the range of jobs. There can be multiple USER job qualifiers on the command.

PROMPT

The **PROMPT** command generates a WTOR.

The normal interface to SOLVE:EPS is with the MVS **MODIFY (F)** command, but the **PROMPT** command allows a WTOR interface. While the WTOR is in effect, the MVS **MODIFY** command interface can still be used.

PROMPT YES | NO

Syntax Description

YES	Specifies that a WTOR be issued to the console. If a WTOR is already outstanding, the command has no effect. YES is the default.
NO	Specifies that the WTOR that is outstanding be removed from the console. If there is no WTOR outstanding, an error message will display.

F EPS,PROMPT

Use **PROMPT NO** to remove the WTOR.

PURGE

The **PURGE** command removes jobs from SOLVE:EPS.

Only held or ready jobs can be purged.

```
PURGE eps_job_id1 [ eps_job_ib2... | THRU eps_job_idn ] [ CLASS class ]
  [ DESTINATION dest ] [ FORMS form ] [ ORPHANS ] [ QUEUE q_name ]
  [ STATUS READY | HOLD | HOLD( OPER ) | HOLD( DISP ) | HOLD( ERROR ) ]
  [ USER user ]
```

Syntax Description

<i>eps_job_idn</i>	ID of the SOLVE:EPS jobs to be purged. You can use the asterisk (*) to purge all jobs from SOLVE:EPS.
THRU	Specifies the range of jobs to be purged. Range is 1 to 99999.
CLASS <i>class</i>	Specifies a class to restrict the range of jobs. There can be multiple CLASS job qualifiers on the command.
DESTINATION <i>dest</i>	Specifies a destination to restrict the range of jobs. This command can have multiple DESTINATION job qualifiers.
FORMS <i>form</i>	Specifies a form to restrict the range of jobs. This command can have multiple FORMS job qualifiers.
ORPHANS	Specifies a restriction of the range of jobs to those on the orphan queue.
QUEUE <i>q_name</i>	Specifies an SOLVE:EPS queue to restrict the range of jobs; if QUEUE is not specified, all queues will be purged. This command can have multiple QUEUE job qualifiers.
STATUS	Introduces a job qualifier for the status. The word that follows indicates which status is being specified. This command can have multiple STATUS job qualifiers. STATUS can be one of the following: • READY Restricts the command to READY jobs • HOLD—Restricts the command to jobs on HOLD status. • HOLD(OPER)—Restricts the command to jobs on operator HOLD status. • HOLD(DISP)—Restricts the command to jobs on disposition HOLD status. • HOLD(ERROR)—Restricts the command to jobs on error HOLD status. • HOLD(EXIT)—Restricts the command to jobs on exit HOLD status. This command purges all jobs that are on error hold or on disposition hold: PURGE * STATUS HOLD(ERR) STATUS HOLD(DISP)
USER <i>user</i>	Specifies a user to restrict the range of jobs. This command can have multiple USER job qualifiers.

REFRESH

The **REFRESH** command refreshes and reloads previously defined PDS members (CONTROLSTRING, SECURITY, and SSISELECT) that have been updated and need SOLVE:EPS to reprocess the members to update internal tables.

If you are specifying multiple members to be refreshed, all members specified will be refreshed before any other work is performed (in other words, no Outprocessors will be given new work).

**REFRESH CONTROLSTRING | CONVERTER | CONVERTSELECTOR |
DATASETEXIT | DISPATCHEXIT | ENVIRONMENTMEMBER | HOSTTABLE |
NJETABLE | SECURITY | SSISELECT | TRANSLATE | UCS *mem_name***

Syntax Description

CONTROLSTRING	Refreshes a control string entry. The alias for CONTROLSTRING is CTLSTR.
CONVERTER	Recompiles the Initialization User Exit for a CONVERTER object. It takes effect immediately, unless a compile error is encountered. If the CONVERTER object does not have an INITEXIT assigned to it, an error results.
CONVERTSELECTOR	Refreshes (loads and compiles) the CONVERTSELECTOR user exit. Unless an error occurs, it takes effect immediately.
DATASETEXIT	Refreshes an EPSUSEX data set User Exit that has been loaded by starting a queue that uses it.
DISPATCHEXIT	Refreshes the Dispatch User Exit code. At least one queue referencing <i>mem_name</i> must be started in order to use the REFRESH DISPATCHEXIT command.
ENVIRONMENTMEMBER	Refreshes the environment variable member used by startable objects. An alias for ENVIRONMENTMEMBER is ENVMEMBER.
HOSTTABLE	Refreshes the host table used by a TCP alternate console.
NJETABLE	Refreshes the NJE Inprocessor selection table.
SECURITY	Refreshes the Inprocessor security table for a BSD-type Inprocessor.
SSISELECT	Refreshes the SSI Inprocessor selection table.
TRANSLATE	Refreshes the SOLVE:EPS language and translation description table. The alias for TRANSLATE is XLATE.
UCS	Refreshes the UCS table used by an NJE- or SSI-type Inprocessor.
<i>mem_name</i>	Name of the previously defined PDS member containing the new data for the specified function. The member must reside in the PDS referenced by the PARMLIB DD statement in the startup JCL. Note: <i>mem_name</i> must be specified or an error occurs.

RELEASE

The **RELEASE** command releases jobs and makes them available for processing by an Outprocessor.

Only held jobs can be released.

```
RELEASE eps_job_id1 [ eps_job_id2... | THRU eps_job_idn ] [ CLASS class ]
  [ DESTINATION dest ] [ FORMS form ] [ ORPHANS ] [ QUEUE q_name ]
  [ STATUS HOLD( OPER ) | HOLD( DISP ) | HOLD( ERROR ) | HOLD( EXIT ) ]
  [ USER user ]
```

Syntax Description

<i>eps_job_idn</i>	ID of the SOLVE:EPS jobs to be released. You can use the asterisk (*) to release all jobs in SOLVE:EPS.
THRU	Specifies the range of jobs to be released, from 1 through 99999.
CLASS <i>class</i>	Specifies a class to restrict the range of jobs. This command can have multiple CLASS job qualifiers.
DESTINATION <i>dest</i>	Specifies a destination to restrict the range of jobs. This command can have multiple DESTINATION job qualifiers.
FORMS <i>form</i>	Specifies a form to restrict the range of jobs. This command can have multiple FORMS job qualifiers.
ORPHANS	Specifies a restriction of the range of jobs to those on the orphan queue.
QUEUE <i>q_name</i>	Specifies an SOLVE:EPS queue to restrict the range of jobs; if QUEUE is not specified, all queues will be released. This command can have multiple QUEUE job qualifiers.
STATUS	Introduces a job qualifier for status. The word that follows indicates which status is being specified. This command can have multiple STATUS job qualifiers. STATUS can be one of the following: • HOLD(OPER)—Restricts the command to jobs on operator HOLD status. • HOLD(DISP)—Restricts the command to jobs on disposition HOLD status. • HOLD(ERROR)—Restricts the command to jobs on error HOLD status. • HOLD(EXIT)—Restricts the command to jobs on exit HOLD status. This command releases all jobs on the HPJET queue that are on operator hold: RELEASE * QUEUE HPJET STATUS HOLD(OPER)
USER <i>user</i>	Specifies a user to restrict the range of jobs. There can be multiple USER job qualifiers on the command.

RESET

The **RESET** command terminates an active job on an Outprocessor and makes it available for later processing.

Only active jobs can be reset.

```
RESET eps_job_id1 [ eps_job_id2... | THRU eps_job_idn ]
    [ CLASS class ] [ DESTINATION dest ] [ FORMS form ]
    [ HOLD | PURGE | READY ] [ QUEUE q_name ] [ USER user ]
```

Syntax Description

<i>eps_job_id</i>	ID of the SOLVE:EPS job to be reset. You can use the asterisk (*) to reset all jobs.
THRU	Range of jobs to be held. Range is 1 to 99999.
CLASS <i>class</i>	Class to restrict the range of jobs. There can be multiple CLASS job qualifiers on the command.
DESTINATION <i>dest</i>	Destination to restrict the range of jobs. This command can have multiple DESTINATION job qualifiers.
FORMS <i>form</i>	Form to restrict the range of jobs. This command can have multiple FORMS job qualifiers.
HOLD	Terminates the job from the Outprocessor and places the job on hold status. Use the RELEASE command to allow the job to be reselected for outprocessing. HOLD is the default.
PURGE	Terminates the job from the Outprocessor and purges the job from the system. The alias for PURGE is DELETE.
READY	Terminates the job from the Outprocessor and places the job in an inactive state, ready to be selected by another Outprocessor. Aliases for READY are INACTIVE, NOHOLD, and RELEASE.
QUEUE <i>q_name</i>	SOLVE:EPS queue to restrict the range of jobs; if QUEUE is not specified, all queues will be reset. This command can have multiple QUEUE job qualifiers.
USER <i>user</i>	User to restrict the range of jobs. This command can have multiple USER job qualifiers.

ROUTE

The **ROUTE** command routes jobs from one queue to another queue. It also checks to make sure a routing queue loop does not occur. This command affects all jobs on the queue waiting to be processed, as well as new jobs being created by an Inprocessor. The effect of this command is a **SET JOB** *eps_job_id* **QUEUE** *queue2* for every new job, processing job, or queued job on *queue1*.

ROUTE *queue1 queue2* [**FORCE**]

Syntax Description

<i>queue1</i>	Name of queue to be routed. It must be defined or an error occurs.
<i>queue2</i>	Name of queue to receive the jobs. It must be defined or an error occurs.
FORCE	Allows multiple hop routing. A queue can be routed to a queue that is routed to another queue. This parameter must be used to allow multiple hops. If the ROUTE command creates multiple hops without this parameter specified, an error message results. If a routing loop occurs an error message results.

To remove routing, issue the **ROUTE** command with *queue1* equal to *queue1*.

For example,

ROUTE *queue1 queue1*

SET ALTERNATECONSOLE

The **SET ALTERNATECONSOLE** command

The alias for SET ALTERNATECONSOLE is MODIFY ALTERNATECONSOLE; aliases for ALTERNATECONSOLE are ACN or ALTCONS.

SET ALTERNATECONSOLE *cons_name* [*PARAMETERS*]

Syntax Description

cons_name Name of the console to be set.

PARAMETERS Any parameters described in [“SET ALTERNATECONSOLE Parameters” on page 1-18](#), following.

SET ALTERNATECONSOLE Parameters

ENVIRONMENTMEMBER

The ENVIRONMENTMEMBER parameter lets you set C runtime environment variables for an alternate console. They are specified by assigning a PARMLIB DD member to the alternate console. The environment settings are specified in the member.

An alias for ENVIRONMENTMEMBER is ENVMEMBER.

This parameter is not dynamic.

SET ALTERNATECONSOLE *cons_name* **ENVIRONMENTMEMBER** *mem_name*

Syntax Description

cons_name Specifies the name of an alternate console.

mem_name Specifies the name of the PARMLIB DD member containing environment variable settings.

HOSTTABLE/NOHOSTTABLE

The HOSTTABLE parameter specifies a member of the PDS pointed to by the PARMLIB DD JCL statement which describes remote hosts and/or users that are allowed access to the alternate console. Read the section “REMOTEIPADDRESS Statement” in “SOLVE:EPS Members” (Chapter 8) of the **SOLVE:EPS Installation and Administration Guide** for information about the host table member.

This parameter is not dynamic.

The default is NOHOSTTABLE.

SET ALTERNATECONSOLE *cons_name* **HOSTTABLE** *mem_name*

Syntax Description

cons_name Name of the alternate console to be set.

mem_name Name of the PDS member to use.

NOHOSTTABLE specifies that the default limitations are placed on connections based on source IP address or host name.

LOGLEVEL

The LOGLEVEL parameter controls the types of messages written to the SOLVE:EPS log for the named alternate console.

This parameter is dynamic.

SET ALTERNATECONSOLE *cons_name* LOGLEVEL
[FATAL | ERROR | WARNING] |
[INFORMATION | NOINFORMATION | DEBUG | NODEBUG]

Syntax Description

<i>cons_name</i>	Name of the alternate console being set.
FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION NOINFORMATION	INFORMATION enables informational messaging (also sets NODEBUG). NOINFORMATION disables informational messaging (also sets NODEBUG).
DEBUG NODEBUG	DEBUG enables debug messaging (also sets INFORMATION). NODEBUG disables debug messaging (also sets INFORMATION).

MESSAGELEVEL

The MESSAGELEVEL parameter controls the types of messages written to the MVS console via WTO for SOLVE:EPS.

This parameter is dynamic.

The alias for MESSAGELEVEL is MSGLEVEL.

SET ALTERNATECONSOLE *cons_name* **MESSAGELEVEL**

[FATAL | ERROR | WARNING] |

[INFORMATION | NOINFORMATION | DEBUG | NODEBUG]

Syntax Description

<i>cons_name</i>	Name of the alternate console being set.
FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION NOINFORMATION	INFORMATION enables informational messaging (also sets NODEBUG). NOINFORMATION disables informational messaging (also sets NODEBUG).
DEBUG NODEBUG	DEBUG enables debug messaging (also sets INFORMATION). NODEBUG disables debug messaging (also sets INFORMATION).

PORT

The PORT parameter specifies the sockets port on which this alternate console listens for connections.

This parameter is not dynamic.

There is no default.

SET ALTERNATECONSOLE *cons_name* **PORT** *port_name*

Syntax Description

<i>cons_name</i>	Name of the alternate console to be set.
<i>port_name</i>	Defines the port number to which remote users connect (0-32767). We recommend a port number greater than 1024.

RESPONSEFILTER/NORESPONSEFILTER

The RESPONSEFILTER parameter specifies the module name of a user exit that filters responses to commands received via the alternate console.

This parameter is not dynamic.

The default is NORESPONSEFILTER.

SET ALTERNATECONSOLE RESPONSEFILTER *mod_name*

A sample T07XXACR is distributed with SOLVE:EPS. See the section [“Response User Exit” on page A-53](#) in the [“User Exits”](#) appendix for details.

NORESPONSEFILTER specifies that there will be no filtering of command responses from the alternate console.

SENDBUFFERSIZE

The SENDBUFFERSIZE parameter specifies the maximum number of bytes of response message text to buffer before a send request is issued. Larger values may improve performance.

This parameter is dynamic.

The default value is 4096. Permissible values are from 256 to 64K. You can use a suffix of K to denote multiples of 1024 bytes.

SET ALTERNATECONSOLE SENDBUFFERSIZE *buf_size*

SESSIONS

The SESSIONS parameter specifies the number of input streams an alternate console can handle simultaneously.

The default for the SESSIONS parameter is 1. The maximum is 250.

This parameter is dynamic.

SET ALTERNATECONSOLE *cons_name* **SESSIONS** *sess_num*

Syntax Description

<i>cons_name</i>	Name of alternate console to be set.
<i>sess_num</i>	Number of sessions. This value must be numeric.

SYSID

The SYSID parameter defines, for an alternate console, the MVS subsystem name of the SOLVE:TCPaccess address space.

This parameter is not dynamic.

SET ALTERNATECONSOLE *cons_name* **SYSID** *sysid*

Syntax Description

<i>cons_name</i>	Name of the alternate console to be set.
<i>sysid</i>	Defines the MVS subsystem name. The default for <i>sysid</i> is ACSS.

TRACEWORD

The TRACEWORD parameter specifies trace settings for the alternate console.

This parameter is dynamic.

The default is 0.

SET ALTERNATECONSOLE *cons_name* **TRACEWORD** *trace_value*

Syntax Description

cons_name Name of the alternate console to be set.

trace_value Value of the TRACEWORD to be set.

Read the chapter [“SOLVE:EPS Operation”](#) for the valid TRACEWORD values.

TRANSLATE

The TRANSLATE parameter of the **SET ALTERNATECONSOLE** command specifies the translation table to use for the alternate console.

The default is ENGLISH.

This parameter is not dynamic.

The alias for TRANSLATE is XLATE.

SET ALTERNATECONSOLE *cons_name* **TRANSLATE** *table_name*

Syntax Description

cons_name Name of the alternate console to be set.

table_name Specifies a translation table name.

UPPERCASE/NOUPPERCASE

The UPPERCASE parameter specifies that commands will be converted to uppercase before being passed to the main SOLVE:EPS task. Commands are normally passed in upper and lower (mixed) case.

This parameter is dynamic.

The default is NOUPPERCASE.

SET ALTERNATECONSOLE UPPERCASE

NOUPPERCASE allows commands to be passed to the main SOLVE:EPS task in mixed case.

SET ALTERNATELOG

The **SET ALTERNATELOG** command

The alias for SET ALTERNATELOG is MODIFY ALTERNATELOG; aliases for ALTERNATELOG are ALG or ALTLOG.

SET ALTERNATELOG *log_name* [*PARAMETERS*]

Syntax Description

log_name Name of the alternate log to be set.

PARAMETERS Any parameters described in [“SET ALTERNATELOG Parameters” on page 1-23](#), following.

SET ALTERNATELOG Parameters

ENVIRONMENTMEMBER

The ENVIRONMENTMEMBER parameter lets you set C runtime environment variables for an alternate log. They are specified by assigning a PARMLIB DD member to the alternate log. The environment settings are specified in the member.

SET ALTERNATELOG *log_name* **ENVIRONMENTMEMBER** *mem_name*

Syntax Description

log_name Specifies the name of an alternate log.

mem_name Specifies the name of the PARMLIB DD member containing environment variable settings.

LOGLEVEL

The LOGLEVEL parameter controls the types of messages written to the SOLVE:EPS log for the named alternate log.

This parameter is dynamic.

SET ALTERNATELOG *log_name* **LOGLEVEL**

[**FATAL** | **ERROR** | **WARNING**] |

[**INFORMATION** | **NOINFORMATION** | **DEBUG** | **NODEBUG**]

Syntax Description

<i>log_name</i>	Name of the alternate log being set.
FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION NOINFORMATION	INFORMATION enables informational messaging (also sets NODEBUG). NOINFORMATION disables informational messaging (also sets NODEBUG).
DEBUG NODEBUG	DEBUG enables debug messaging (also sets INFORMATION). NODEBUG disables debug messaging (also sets INFORMATION).

MESSAGEFILTER/NOMESSAGEFILTER

The MESSAGEFILTER parameter specifies the module name of a user exit that filters log messages sent via the alternate log.

This parameter is not dynamic.

Aliases for MESSAGEFILTER are MSGFILTER and RESPONSEFILTER.

The default is NORESPONSEFILTER.

SET ALTERNATELOG MESSAGEFILTER *mod_name*

A sample T07XXALE is distributed with (Continued). See the section [“Response User Exit” on page A-53](#) in the [“User Exits”](#) appendix for details.

NOMESSAGEFILTER specifies that there will be no filtering of log messages sent via the alternate log.

Aliases for NOMESSAGEFILTER are NOMSGFILTER and NORESPONSEFILTER.

MESSAGELEVEL

The MESSAGELEVEL parameter controls the types of messages written to the MVS console via WTO for SOLVE:EPS.

This parameter is dynamic.

The alias for MESSAGELEVEL is MSGLEVEL.

```
SET ALTERNATELOG log_name MESSAGELEVEL
  [ FATAL | ERROR | WARNING ] |
  [ INFORMATION | NOINFORMATION | DEBUG | NODEBUG ]
```

Syntax Description

log_name	Name of the alternate log being set.
FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION NOINFORMATION	INFORMATION enables informational messaging (also sets NODEBUG). NOINFORMATION disables informational messaging (also sets NODEBUG).
DEBUG NODEBUG	DEBUG enables debug messaging (also sets INFORMATION). NODEBUG disables debug messaging (also sets INFORMATION).

RETRYINTERVAL

The RETRYINTERVAL parameter specifies the number of seconds the alternate log is to wait between attempts to establish a PC connection with an e-Control address space.

This parameter is dynamic.

```
RETRYINTERVAL time
```

Syntax Description

time	Seconds between connection attempts. Range is between 5 and 300 seconds. Default: 30
------	---

SYSID

The SYSID parameter defines, for an alternate log, the MVS subsystem name of the e-Control address space.

This parameter is not dynamic.

SET ALTERNATELOG *log_name* **SYSID** *sysid*

Syntax Description

log_name Name of the alternate log to be set.

sysid Defines the MVS subsystem name. The default for *sysid* is OMXC.

TRACEWORD

The TRACEWORD parameter specifies trace settings for the alternate log.

This parameter is dynamic.

The default is 0.

SET ALTERNATELOG *log_name* **TRACEWORD** *trace_value*

Syntax Description

log_name Name of the alternate log to be set.

trace_value Value of the TRACEWORD to be set.

Read the chapter [“SOLVE:EPS Operation”](#) for the valid TRACEWORD values.

SET CONVERTER

The **SET CONVERTER** command modifies a defined CONVERTER object. The changes take effect immediately, unless an error is encountered.

The alias for SET CONVERTER is MODIFY CONVERTER.

SET CONVERTER *conv_name* **INITMODULE** *mod_name*
[**INITEXIT** *mem_name*]

Syntax Description

conv_name Name of the converter, up to 32 non-blank characters.

mod_name Name of the converter initialization module.

mem_name Name of an optional converter Initialization User Exit written in EPSUSEX.

See [“DEFINE ALTERNATELOG” on page 1-6](#) for an explanation of the keyword options.

To remove a converter Initialization User Exit from a CONVERTER object, use the NOINITEXIT parameter:

SET CONVERTER *conv_name* **NOINITEXIT**

SET SOLVE:EPS

Use the **SET EPS** command to change the behavior of the main component of SOLVE:EPS. This component manages all the objects within SOLVE:EPS.

The alias for **SET EPS** is **MODIFY EPS**.

SET EPS [*PARAMETERS*]

Syntax Description

PARAMETERS Any parameters described in the following section, [“SET EPS Parameters” on page 1-28](#).

SET EPS Parameters

This section describes the valid parameters for the **SET EPS** command.

LOGLEVEL

The LOGLEVEL parameter controls the types of messages written to the log for SOLVE:EPS.

This parameter is dynamic.

SET EPS LOGLEVEL [**FATAL** | **ERROR** | **WARNING**] |
[**INFORMATION** | **NOINFORMATION** | **DEBUG** | **NODEBUG**]

Syntax Description

FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION NOINFORMATION	INFORMATION enables informational messaging (also sets NODEBUG). NOINFORMATION disables informational messaging (also sets NODEBUG).
DEBUG NODEBUG	DEBUG enables debug messaging (also sets INFORMATION). NODEBUG disables debug messaging (also sets INFORMATION).

LOWERCASEINDICATOR/*NOLOWERCASEINDICATOR

The LOWERCASEINDICATOR parameter lets you specify any character to be used as an indicator for a parameter string value to be translated to lowercase. When this character is the first character of a parameter value or object name in an SOLVE:EPS command, the remaining characters will be translated to lower case with the indicator character removed. This allows commands to be entered from the MVS console to contain lower case object names and/or parameter values without having to setup a special exec.

This parameter is dynamic.

SET EPS LOWERCASEINDICATOR *char* | NOLOWERCASEINDICATOR

In the startup parameters, use the following:

SET EPS LOWERCASEINDICATOR !

DEFINE QUEUE *any*

Now a command at the MVS console can be entered as shown below, which will allow the lower case "queue any" to display:

SHOW QUEUE !ANY

MESSAGELEVEL

The MESSAGELEVEL parameter controls the types of messages for SOLVE:EPS written to the MVS console via the WTO interface.

This parameter is dynamic.

The alias for MESSAGELEVEL is MSGLEVEL.

**SET EPS MESSAGELEVEL [FATAL | ERROR | WARNING] |
[INFORMATION | NOINFORMATION | DEBUG | NODEBUG]**

Syntax Description

FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION NOINFORMATION	INFORMATION enables informational messaging (also sets NODEBUG). NOINFORMATION disables informational messaging (also sets NODEBUG).
DEBUG NODEBUG	DEBUG enables debug messaging (also sets INFORMATION). NODEBUG disables debug messaging (also sets INFORMATION).

UPPERCASE/NOUPPERCASE

The UPPERCASE parameter specifies how messages will be written to the MVS console. Messages are normally written in upper and lower (mixed) case. The UPPERCASE parameter causes the entire message to be written in upper case.

This parameter is dynamic.

The default is NOUPPERCASE.

Note: Some of the parameters in SOLVE:EPS are specified in lower case and writing messages that contain these parameters to the console in upper case may cause confusion.

SET EPS UPPERCASE

The NOUPPERCASE parameter sets the writing of messages in the log to mixed case.

SET EPS NOUPPERCASE**TRACEWORD**

In conjunction with support for the Dispatch User Exit written in EPSUSEX, a TRACEWORD bit setting for SOLVE:EPS has been added.

SET EPS TRACEWORD 40000000

This command requests a dump of the byte codes generated when Dispatch User Exit code is compiled. The results appear only in the SOLVE:EPS log.

TRACEWORD specifies a value in hexadecimal. To set other TRACEWORD options in addition to this one, OR their values with it.

Normally it will only be necessary to trace EPSUSEX byte codes when asked for by Customer Support.

SET INPROCESSOR

The **SET INPROCESSOR** command changes parameter values associated with an Inprocessor.

These parameters are also valid on a **DEFINE INPROCESSOR** command.

The alias for **SET INPROCESSOR** is **MODIFY INPROCESSOR**.

SET INPROCESSOR *inp_name* [*PARAMETERS*]

Syntax Description

inp_name Name of the Inprocessor to be set. The Inprocessor must have been previously defined with a **DEFINE INPROCESSOR** command.

PARAMETERS Any of the parameters described in the following sections,
 [“SET INPROCESSOR Parameters” on page 1-31](#),
 [“Job Defaults” on page 1-36](#),
 [“BSD-Specific Parameters” on page 1-40](#),
 [“NJE-Specific Parameters” on page 1-41](#),
 [“SSI-Specific Parameters” on page 1-45](#), and
 [“VPRT-Specific Parameters” on page 1-46](#).

Note: Not all of the parameters are applicable to all types of Inprocessors. If a parameter is specified and is not applicable to the type of Inprocessor used, an error message will be issued.

SET INPROCESSOR Parameters

This section describes the valid parameters for the **SET INPROCESSOR** command.

DEFAULT/ NODEFAULT

The **DEFAULT** parameter makes an Inprocessor model the default for subsequent **DEFINE INPROCESSOR** commands.

A **LIKE** parameter need not be specified on subsequent **DEFINE INPROCESSOR** commands to have the new definition assume the parameters of the **DEFAULT** definition. If a default has already been set, this replaces that setting.

This parameter is dynamic.

SET INPROCESSOR *model_name* **DEFAULT** | **NODEFAULT**

Syntax Description

model_name Name of an Inprocessor model. If the named Inprocessor is not a model, an error message results.

The **NODEFAULT** parameter removes the **DEFAULT** attribute from an Inprocessor model. Subsequent **DEFINE INPROCESSOR** commands will have no default model to alter the internal Inprocessor defaults.

SET INPROCESSOR *model_name* **NODEFAULT**

ENVIRONMENTMEMBER

The new ENVIRONMENTMEMBER parameter makes it possible to set C runtime environment variables for BSD Inprocessors and Outprocessors. They are specified by assigning a PARMLIB DD member to the Inprocessors and Outprocessors. The environment settings are specified in the member.

SET INPROCESSOR *inp_name* **ENVIRONMENTMEMBER** *mem_name*

Syntax Description

<i>inp_name</i>	Specifies the name of a BSD Inprocessor.
<i>mem_name</i>	Specifies the name of the PARMLIB DD member containing environment variable settings.

LOGLEVEL

The LOGLEVEL parameter controls the types of messages written to the SOLVE:EPS log for the named Inprocessor.

This parameter is dynamic.

SET INPROCESSOR *inp_name* **LOGLEVEL** [**FATAL** | **ERROR** | **WARNING**] |
[**INFORMATION** | **NOINFORMATION** | **DEBUG** | **NODEBUG**]

Syntax Description

<i>inp_name</i>	Name of the Inprocessor being set.
FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION NOINFORMATION	INFORMATION enables informational messaging (also sets NODEBUG). NOINFORMATION disables informational messaging (also sets NODEBUG).
DEBUG NODEBUG	DEBUG enables debug messaging (also sets INFORMATION). NODEBUG disables debug messaging (also sets INFORMATION).

MESSAGELEVEL

The MESSAGELEVEL parameter controls the types of messages written to the MVS console via WTO for SOLVE:EPS.

This parameter is dynamic.

The alias for MESSAGELEVEL is MSGLEVEL.

SET INPROCESSOR *inp_name* MESSAGELEVEL

[FATAL | ERROR | WARNING] |

[INFORMATION | NOINFORMATION | DEBUG | NODEBUG]

Syntax Description

<i>inp_name</i>	Inprocessor being set.
FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION NOINFORMATION	INFORMATION enables informational messaging (also sets NODEBUG). NOINFORMATION disables informational messaging (also sets NODEBUG).
DEBUG NODEBUG	DEBUG enables debug messaging (also sets INFORMATION). NODEBUG disables debug messaging (also sets INFORMATION).

QUEUEDEFAULT

The QUEUEDEFAULT parameter defines the queue on which the Inprocessor will place a job if a queue cannot be determined as documented in “System Architecture” (Chapter 2) of the ***SOLVE:EPS Installation and Administration Guide***.

This parameter is not dynamic.

SET INPROCESSOR *inp_name* QUEUEDEFAULT *q_name*

Syntax Description

<i>inp_name</i>	Name of the Inprocessor to be set.
<i>q_name</i>	Queue object to be used as a default if the incoming job does not provide a way of determining the queue.

PRIORITY/NOPRIORITY

The PRIORITY parameter lets you set a priority for a newly created job.

SET INPROCESSOR *inp_name* **PRIORITY** *priority*

Syntax Description

<i>inp_name</i>	Name of the Inprocessor to set the priority.
<i>priority</i>	Value of the priority to be set for new jobs being created. It ranges from 1 to 999, with 1 being the highest and 999 being the lowest. Default is NOPRIORITY.

SESSIONS

The SESSIONS parameter specifies the number of input streams an Inprocessor can handle simultaneously.

This parameter varies according to type. For example, the NJE and SSI Inprocessors can have only one session, the BSD Inprocessor can vary the number of sessions, the VPRT Inprocessor can handle any number of sessions. The BSD Inprocessor uses this parameter as the number of simultaneous inbound connections that can be active at any one time.

The default for the SESSIONS parameter is 1. A specification of 0 calls for the maximum number of sessions. For a BSD Inprocessor, the maximum is 256.

This parameter is dynamic.

SET INPROCESSOR *inp_name* **SESSIONS** *sess_num*

Syntax Description

<i>inp_name</i>	Inprocessor to be set.
<i>sess_num</i>	Number of sessions. This value must be numeric.

TRACEWORD

The TRACEWORD parameter specifies trace settings for the Inprocessor.

This parameter is dynamic.

SET INPROCESSOR *inp_name* **TRACEWORD** *trace_value*

Syntax Description

<i>inp_name</i>	Name of the Inprocessor to be set.
<i>trace_value</i>	Value of the TRACEWORD to be set.

Read the chapter [“SOLVE:EPS Operation”](#) for the valid TRACEWORD values.

TRANSLATE

The TRANSLATE parameter of the **SET INPROCESSOR** command specifies the desired translation for jobs generated by the Inprocessor.

TRANSLATE indicates the desired language code set and encoding for SOLVE:EPS jobs. Translation changes the encoding of a job from EBCDIC to ASCII and from ASCII to EBCDIC. If NONE is specified, the data is not translated.

Refer to the translate member of the SOLVE:EPS PARMLIB DD for the supported translations. This member also specifies the default translation.

The default is UNSPECIFIED.

This parameter is not dynamic.

The alias for TRANSLATE is XLATE.

Note: While TRANSLATE can be specified for Inprocessors, it is recommended that Inprocessors specify translation through their respective tables. See Chapter 11, "SOLVE:EPS Members."

SET INPROCESSOR *inp_name* TRANSLATE UNSPECIFIED | *lang_name* | NONE

Syntax Description

<i>inp_name</i>	Specifies the name of the Inprocessor to be set.
UNSPECIFIED	Specifies that there is no translation specified for this Inprocessor. Translation is determined by the queue and the translate member of the PARMLIB DD. This is the default. The alias for UNSPECIFIED is UNSET.
<i>lang_name</i>	Specifies the name of the desired language translation. This value must exist in the translate member of the PARMLIB DD.
NONE	Specifies that no translation be used for jobs generated by the Inprocessor.

Job Defaults

This section describes parameters for the **SET INPROCESSOR** command that relate to jobs. Use these parameters to supply job attributes not provided by the input stream or the queue.

The prefixes described here can be added to any of the job defaults parameters for the **SET INPROCESSOR** command:

- **!** – An exclamation point keyword preceding a job default parameter specifies to remote the remotely received value.
- **@** – An at sign keyword specifies to remove the extended class value provided on the extended class -C remote lpr command.
- **#** – A number sign keyword specifies to remove the lpr control file value that is sent by the remote lpr command.

RPRINTER *printer_name* **QUEUE** *njeout* **#JOBNAME** **@ACCOUNT** **ACCOUNT 8000**
JOBNAME *job_name*

Note: National characters are:

\$ # @

ACCOUNT/NOACCOUNT

ACCOUNT specifies the accounting information for the job. The field can be from 1 to 64 nonblank characters.

NOACCOUNT explicitly specifies that no **ACCOUNT** information is specified for the job.

CHARn/NOCHARn

CHARn specifies the first character arrangement table for printing the job on a 3800 printing subsystem. This applies only for a job that will eventually be printed on a 3800. Otherwise, it has no effect. Each table name can be from 1 to 4 alphanumeric or national characters. There can be up to four character arrangement tables, (**CHAR1**, **CHAR2**, **CHAR3**, and **CHAR4**).

NOCHARn explicitly specifies that no **CHAR** is specified for the job.

CLASS/NOCLASS

CLASS specifies the output class for the job. **CLASS** is one character, from A through Z or from 0 through 9. The attributes of each class are defined by the destination system.

NOCLASS explicitly specifies that no **CLASS** is specified for the job.

DESTINATION/NODESTINATION

DESTINATION specifies the JES **DESTINATION** of the job on the destination system.

NODESTINATION explicitly specifies that no **DESTINATION** is specified for the job.

Note: If you code **DEST=nodename** with the .userid extension concurrently with **WRITER=name** an MVS error message will result.

FCB/NOFCB

FCB specifies the Forms Control Buffer (FCB) image to use to guide printing a job on a JES printer. The name can be from 1 to 4 alphanumeric or national characters.

NOFCB explicitly specifies that no Forms Control Buffer is specified for the job.

FLASH /NOFLASH

FLASH specifies the forms overlay to be used in printing the job on a 3800 printing subsystem. The name can be from 1 to 4 alphanumeric or national characters.

NOFLASH explicitly specifies that no forms overlay is specified for the job.

FORMDEF/NOFORMDEF

FORMDEF specifies the name of a library member used by Print Services Facility (PSF) for printing on a page mode printer. The name can be from 1 to 6 alphanumeric or national characters.

NOFORMDEF explicitly specifies that no FORMDEF library member is specified for the job.

FORMS/NOFORMS

FORMS specifies the job's print forms. FORMS can be from 1 to 4 alphanumeric or national characters.

NOFORMS explicitly specifies that no FORMS are specified for the job.

HOLD/NOHOLD

HOLD specifies to hold the job.

NOHOLD explicitly specifies that there is no HOLD on the job.

INDENT/NOINDENT

INDENT specifies that the job data should be shifted right the specified number of columns. The value can be from 0 to 65535.

NOINDENT explicitly specifies that there is to be no indentation for the job.

JOBNAME/NOJOBNAME

JOBNAME specifies the job name that the job will have if one has not already been determined by the Inprocessor. The job name can be from 1 to 8 alphanumeric and national characters.

NOJOBNAME explicitly specifies that no JOBNAME is specified for the job.

JOBPROGRAMMER/NOJOBPROGRAMMER

JOBPROGRAMMER specifies the programmer's name field of a job card which can help identify the person responsible for the job. The field can be from 1 to 20 nonblank characters.

NOJOBPROGRAMMER explicitly specifies that no JOBPROGRAMMER is specified for the job.

The alias for JOBPROGRAMMER is JOBPGRM.

JOBROOM/NOJOBROOM

JOBROOM specifies the room number for the job. The room number can be from 1 to 8 alphanumeric or national characters.

NOJOBROOM explicitly specifies that no JOBROOM number is specified for the job.

LINECOUNT/NOLINECOUNT

LINECOUNT specifies the maximum number of lines that JES2 is to print on each output page for the job. The number can be from 0 to 254. If specified as 0, JES2 does not eject to a new page when the number of output lines exceeds the page limit that the installation specified during JES2 initialization.

NOLINECOUNT explicitly specifies that no LINECOUNT is specified for the job.

The alias for LINECOUNT is LINECT.

Note: This job default command is for JES2 only.

NOTIFYUSER/NONOTIFYUSER

NOTIFYUSER specifies the user name of a user to be notified when the job completes printing on a JES printer. The name can be from 1 to 7 alphanumeric or national characters.

NONOTIFYUSER explicitly specifies that no NOTIFYUSER is specified for the job.

PAGEDEF/NOPAGEDEF

PAGEDEF specifies the name of a library member used by Print Services Facility (PSF) for printing on a page mode printer. The name can be from 1 to 6 alphanumeric or national characters.

NOPAGEDEF explicitly specifies that no PAGEDEF is specified for the job.

PRMODE/NOPRMODE

PRMODE specifies the processing mode required to print the job.

- LINE specifies to schedule the job to a line mode printer.
- PAGE specifies to schedule the job to a page mode printer.

NOPRMODE explicitly specifies that no PRMODE is specified for the job.

TRC/NOTRC

TRC specifies to a 3800 printing subsystem that each logical record contains a Table Reference Character (TRC). The TRC identifies which character arrangement table in the CHARx parameters is to be used to print the line.

Using SET INPROCESSOR TRC is the same as using the JCL parameter DD DCB=OPTCD=J.

NOTRC explicitly specifies that no TRC is specified for the job.

UCS/NOUCS

UCS specifies the Universal Character Set (UCS) to be used to print the job on a JES printer. The name can be from 1 to 4 alphanumeric or national characters.

NOUCS explicitly specifies that no UCS is specified for the job.

WIDTH/NOWIDTH

WIDTH specifies the width of the job data. It is the maximum line length before a new line is forced. The value can be from 0 to 65535.

NOWIDTH explicitly specifies that there is no WIDTH limitation for the job.

WRITER/NOWRITER

WRITER names an external writer. The name can be from 1 to 8 alphanumeric characters.

NOWRITER explicitly specifies that no WRITER is specified for the job.

Note: If you code DEST=nodename with the .userid extension concurrently with WRITER=name an MVS error message will result.

BSD-Specific Parameters

This section describes the valid BSD-specific parameters for the **SET INPROCESSOR** command.

PORT

The PORT parameter specifies the port for the BSD LPD server for this Inprocessor.

This parameter is not dynamic.

The default is 515, which is the standard BSD LPD well-known port.

SET INPROCESSOR *inp_name* **PORT** *port_name*

Syntax Description

inp_name Name of the Inprocessor to be set.

port_name Defines the BSD LPD server well-known port (0-32767).

SECURITY/NOSECURITY

The SECURITY parameter specifies a member of the PDS pointed to by the PARMLIB DD JCL statement that describes remote hosts and/or users that are allowed access to the LPD server of SOLVE:EPS.

This parameter is not dynamic.

If this parameter is not specified, the default is member T07BSD.

Read “LPD Server Security” in “SOLVE:EPS Members” (Chapter 8) of the ***SOLVE:EPS Installation and Administration Guide*** for details about the security member.

SET INPROCESSOR *inp_name* **SECURITY** *mem_name*

Syntax Description

inp_name Name of the Inprocessor to be set.

mem_name Name of the PDS member to use.

NOSECURITY specifies that no security table be used. If no security table is used the default is that no access is allowed.

SYSID

The SYSID parameter defines, for a BSD Inprocessor, the MVS subsystem name of the SOLVE:TCPaccess address space. The SYSID parameter is used only by the BSD Inprocessor.

This parameter is not dynamic.

SET INPROCESSOR *inp_name* **SYSID** *sysid*

Syntax Description

inp_name Name of the Inprocessor to be set.

sysid Defines the MVS subsystem name. The default for *sysid* is ACSS.

NJE-Specific Parameters

The NJE parameters are used to define an NJE interface with JES. The interface is an LU0 connection via VTAM. All jobs to be sent to SOLVE:EPS will be sent via the JCL parameter DESTINATION(*node.dest*) or its equivalent. The job will be placed on the SOLVE:EPS queue that matches dest. If there is no matching queue entry defined in SOLVE:EPS, the job will be placed on an “orphan” queue or the queue specified by the SET INPROCESSOR queue default parameter.

JESAPPLID

The JESAPPLID parameter specifies the VTAM LU name of the JES subsystem for an NJE Inprocessor.

This parameter is not dynamic.

The alias for JESAPPLID is JESACB.

SET INPROCESSOR *inp_name* **JESAPPLID** *jesappl*

Syntax Description

<i>inp_name</i>	Inprocessor to be set.
<i>jesappl</i>	VTAM APPLID of the JES host system and provides the network node name of the JES subsystem to which SOLVE:EPS makes the underlying VTAM connection.

There is no default.

JESNODE

The JESNODE parameter specifies the JES NJE node name for an NJE Inprocessor.

This parameter is not dynamic.

SET INPROCESSOR *inp_name* **JESNODE** *jesnode*

Syntax Description

<i>inp_name</i>	Name of the Inprocessor to be set.
<i>jesnode</i>	NJE node name of the JES host system for the SOLVE:EPS-to-JES NJE session. When omitted, <i>jesnode</i> defaults to the <i>jesappl</i> value.

Read “Configuring JES” (Chapter 5) of the **SOLVE:EPS Installation and Administration Guide** for information about JES.

JESPASSWORD/NOJESPASSWORD

The JESPASSWORD parameter specifies the password for the JES NJE node for an NJE Inprocessor.

This parameter is not dynamic.

The alias for JESPASSWORD is JESPSWD.

SET INPROCESSOR *inp_name* **JESPASSWORD** *jespswd*

Syntax Description

inp_name Inprocessor to be set.

jespswd 1- to 8-character password to be used when the node is signing on to the JES node.

NOJESPASSWORD specifies that there is no password.

NJEAPPLID

The NJEAPPLID parameter specifies the VTAM ACB name for this NJE Inprocessor.

This parameter is not dynamic.

The alias for NJEAPPLID is NJEACB.

SET INPROCESSOR *inp_name* **NJEAPPLID** *njeacb*

Syntax Description

inp_name Inprocessor to be set.

njeacb VTAM ACB name for SOLVE:EPS. This value should match the ACB name value on the APPL statement for the SOLVE:EPS application in a VTAM definitions file (usually a member of SYS1.VTAMLST). SOLVE:EPS uses this value as a parameter on the OPEN request to VTAM.

NJENODE

The NJENODE parameter specifies the NJE node name for this NJE Inprocessor.

This parameter is not dynamic.

SET INPROCESSOR *inp_name* **NJENODE** *nje_node*

Syntax Description

inp_name Name of the Inprocessor to be set.

nje_node NJE node name of SOLVE:EPS for the SOLVE:EPS-to-JES NJE session.

The *nje_node* value must match the name value on the NODE statement for SOLVE:EPS in the JES2 parameter file (usually the JES2PARM member of SYS1.PARMLIB). SOLVE:EPS uses NJENODE during NJE handshaking with I and J signon records (NJE path management signon function).

For JES3, match the name parameter on the NJERMT statement of the JES3 INISH deck. Read "Configuring JES" (Chapter 5) of the ***SOLVE:EPS Installation and Administration Guide*** for information about JES.

NJEPARTNER

The NJEPARTNER parameter specifies the type of NJE partner to which the **SET INPROCESSOR** command is communicating. This allows the NJE Inprocessor and Outprocessor to make better decisions on how to map NJE header fields.

The NJEPARTNER parameter takes one of the following values:

- JES2
- JES3
- POWER
- RSCS

SET INPROCESSOR *inp_name* **NJEPARTNER** JES3

The default value for NJEPARTNER is JES2.

NJEPASSWORD/NONJEPASSWORD

The NJEPASSWORD parameter specifies the ACB password for the NJEAPPLID for this NJE Inprocessor.

This parameter is not dynamic.

The alias for NJEPASSWORD is NJEPSWD.

SET INPROCESSOR *inp_name* **NJEPASSWORD** *njepswd*

Syntax Description

<i>inp_name</i>	Name of the Inprocessor to be set.
<i>njepswd</i>	Specifies the password for NJEACB.

This value must match the PRTCT value, if one is specified, on the APPL statement for SOLVE:EPS ACB. If PRTCT is not specified, NJEPSWD must be omitted or a VTAM error message occurs when the Inprocessor is started.

NJETABLE

The NJETABLE parameter specifies the member name of a table that lets you override NJE job fields when a job is being received by the NJE Inprocessor.

There is no default for the NJETABLE parameter.

SET INPROCESSOR *inp_name* **NJETABLE** *mem_name*

Syntax Description

<i>inp_name</i>	Name of the Inprocessor to set.
<i>mem_name</i>	Name of the PARMLIB DD member that contains the NJE override table.

UCSTABLE

The UCSTABLE parameter of the **SET INPROCESSOR** command specifies the PARMLIB DD member to use to map UCS values to translations, and/or lpr filters.

When an Inprocessor receives a job that has a UCS field set to one of the values in the UCSTABLE member, the corresponding translation, conversion, and lpr filter options override any other specifications.

This parameter is optional and has no default.

This parameter is not dynamic.

SET INPROCESSOR *inp_name* **UCSTABLE** *tbl_name*

Syntax Description

<i>inp_name</i>	Name of the Inprocessor to be set.
<i>tbl_name.</i>	Name of the table (PARMLIB DD member) containing the UCS mappings

SSI-Specific Parameters

This section describes SSI-specific parameters for the **SET INPROCESSOR** command.

SSISELECT

The SSISELECT parameter specifies the type of selection to be done by an SSI Inprocessor. An SSI Inprocessor (which uses the MVS subsystem interface) can select jobs from the JES spool based on class, writer, form, and/or destination. This parameter specifies the member name in the PDS pointed to by the PARMLIB DD JCL statement which contains the selection criteria.

This parameter is not dynamic.

Read “SSI Selection Criteria” in “SOLVE:EPS Members” (Chapter 8) of the ***SOLVE:EPS Installation and Administration Guide*** for more information about the member format.

SET INPROCESSOR SSISELECT *mem_name*

Syntax Description

<i>mem_name</i>	Member name from T07PARM that contains the selection criteria. The default for <i>mem_name</i> is T07SSI.
-----------------	---

UCSTABLE

The UCSTABLE parameter of the **SET INPROCESSOR** command specifies the PARMLIB DD member to use to map UCS values to translations, and/or lpr filters.

When an Inprocessor receives a job that has a UCS field set to one of the values in the UCSTABLE member, the corresponding translation, conversion, and lpr filter options override any other specifications.

This parameter is optional and has no default.

This parameter is not dynamic.

SET INPROCESSOR *inp_name* **UCSTABLE** *tbl_name*

Syntax Description

<i>inp_name</i>	Name of the Inprocessor to be set.
<i>tbl_name.</i>	Name of the table (PARMLIB DD member) containing the UCS mappings

VPRT-Specific Parameters

This section describes VPRT-specific parameters for the **SET INPROCESSOR** command.

LOGICALUNITTABLE

The LOGICALUNITTABLE parameter specifies the member name that contains LUNAMES (VTAM ACBs) and QUEUE relationships.

The alias for LOGICALUNITTABLE is LUTABLE.

Read “Logical Unit Table” in “SOLVE:EPS Members” (Chapter 8) in the ***SOLVE:EPS Installation and Administration Guide*** for more information.

SET INPROCESSOR *inp_name* **LOGICALUNITTABLE** *mem_name*

Syntax Description

<i>inp_name</i>	Name of the Inprocessor to be set.
<i>mem_name</i>	Name of the member from T07PARM that contains the LU names. The default for <i>mem_name</i> is T07VPT.

NJENODE

The NJENODE parameter lets you group files and/or write the file to JES spool.

This parameter is not dynamic.

SET INPROCESSOR *vpert_inp* **NJENODE** *nje_node*

Syntax Description

<i>vpert_inp</i>	Name of the VPRT Inprocessor to be set.
<i>nje_node</i>	NJE node name of SOLVE:EPS for the SOLVE:EPS-to-JES NJE session.

If this parameter is set, the VPRT Inprocessor will write its files to the JES spool.

This value must match the name value on the NODE statement for SOLVE:EPS in the JES2 parameter file (usually the JES2PARM member of SYS1.PARMLIB). SOLVE:EPS uses NJENODE during NJE handshaking with I and J signon records (NJE path management signon function).

For JES3, match the name parameter on the NJERMT statement of the JES3 INISH deck. Read “Configuring JES” (Chapter 5) in the ***SOLVE:EPS Installation and Administration Guide*** for information about JES.

OPENJOBLIMIT

The OPENJOBLIMIT parameter limits the number of concurrently open SOLVE:EPS jobs for VPRT Inprocessors.

The default value given to VPRT Inprocessors with no OPENJOBLIMIT specification is 10 concurrent SOLVE:EPS jobs.

Note: The OPENJOBLIMIT setting does not affect the number of concurrently active VTAM sessions allowed by the Inprocessor; it only limits the number of those sessions that can be creating SOLVE:EPS jobs simultaneously.

When a VTAM session requests creation of a SOLVE:EPS job, the VPRT Inprocessor checks the OPENJOBLIMIT setting. If opening a new SOLVE:EPS job would exceed the setting, the request is suspended within the Inprocessor. The session resumes when other sessions close the SOLVE:EPS jobs they are creating and the number of concurrently active SOLVE:EPS jobs drops below the OPENJOBLIMIT setting.

This parameter is dynamic.

SET INPROCESSOR *inp_name* **OPENJOBLIMIT** *max_num*

Syntax Description

<i>inp_name</i>	Specifies the name of the VPRT Inprocessor to set.
<i>max_num</i>	Specifies the maximum number of open SOLVE:EPS jobs allowed for the Inprocessor (valid values range from 1 through 1024).

SET JOB

The **SET JOB** command changes specific attributes about a job.

The alias for **SET JOB** is MODIFY JOB.

Note: The job must be inactive (in a Hold or Ready state) in order for the **SET JOB** command to be effective.

SET JOB *eps_job_id1* [*eps_job_idn* ... | **THRU** *eps_job_idn*] [*PARAMETERS*]

Syntax Description

- | | |
|--------------------|--|
| <i>eps_job_idn</i> | ID of the SOLVE:EPS job to be set. If job ID does not exist within the SOLVE:EPS system, an error will result. |
| THRU | Specifies the range of jobs to be set. Range is 1 to 99999. |
| <i>PARAMETERS</i> | Any parameters described in the following section, “SET JOB Parameters” on page 1-48 , and “Job Defaults” on page 1-49 . |

SET JOB Parameters

This section describes the valid parameters for the **SET JOB** command.

DISPOSITION

The DISPOSITION parameter specifies the disposition of the job after it has been successfully sent via an Outprocessor.

SET JOB *eps_job_id* **DISPOSTION PURGE | HOLD**

Syntax Description

- | | |
|-------------------|--|
| <i>eps_job_id</i> | ID of the SOLVE:EPS job to be set. |
| PURGE | Specifies the job will be deleted when normal outprocessing is complete.
The alias for PURGE is DELETE. |
| HOLD | Specifies the job will be held in the SOLVE:EPS spool when normal outprocessing is complete. |

PRIORITY

The PRIORITY parameter changes the jobs priority within the queue.

SET JOB *eps_job_id* **PRIORITY** *priority*

Syntax Description

<i>eps_job_id</i>	ID of the SOLVE:EPS job to be set.
<i>priority</i>	Specifies the new priority for the job. The priority value ranges from 1 to 999, with 1 being the highest priority and 999 the lowest.

QUEUE

The QUEUE parameter specifies the queue onto which the job should be placed. If the queue does not exist, a warning will result.

SET JOB *eps_job_id* **QUEUE** *q_name*

Syntax Description

<i>eps_job_id</i>	ID of the SOLVE:EPS job to be set.
<i>q_name</i>	Name of the new queue for the job. If the queue does not exist, the job will be considered orphaned. An orphaned job cannot be processed by an Outprocessor.

Job Defaults

This section describes parameters for the **SET JOB** command. Use these parameters to supply job attributes not provided by the input stream or the queue.

Note: National characters are:

\$ # @

CHARn/NOCHARn

CHAR1 specifies the first character arrangement table for printing the job on a 3800 printing subsystem. This applies only for a job that will eventually be printed on a 3800. Otherwise, it has no effect. Each table name can be from 1 to 4 alphanumeric or national characters. There can be up to four character arrangement tables, (CHAR1, CHAR2, CHAR3, and CHAR4).

NOCHARn explicitly specifies that no CHAR is specified for the job.

CLASS/NOCLASS

CLASS specifies the output class for the job. CLASS is one character, from A through Z or from 0 through 9. The attributes of each class are defined by the destination system.

NOCLASS explicitly specifies that no CLASS is specified for the job.

DESTINATION/NODESTINATION

DESTINATION specifies the JES DESTINATION of the job on the destination system.

NODESTINATION explicitly specifies that no DESTINATION is specified for the job.

Note: If you code DEST=nodename with the .userid extension concurrently with WRITER=name an MVS error message will result.

FCB/NOFCB

FCB specifies the Forms Control Buffer (FCB) image to be used to guide printing of the job on a JES printer. The name can be from 1 to 4 alphanumeric or national characters.

NOFCB explicitly specifies that no FCB is specified for the job.

FLASH/NOFLASH

FLASH specifies the forms overlay to be used in printing the job on a 3800 printing subsystem. The name can be from 1 to 4 alphanumeric or national characters.

NOFLASH explicitly specifies that no FLASH is specified for the job.

FORMDEF/NOFORMDEF

FORMDEF specifies the name of a library member used by Print Services Facility (PSF) for printing on a page mode printer. The name can be from 1 to 6 alphanumeric or national characters.

NOFORMDEF explicitly specifies that no FORMDEF library member is specified for the job.

FORMS/NOFORMS

FORMS specifies the job's print forms. FORMS can be from 1 to 4 alphanumeric or national characters.

NOFORMS explicitly specifies that no FORMS are specified for the job.

HOLD/NOHOLD

HOLD specifies to hold the job.

NOHOLD explicitly specifies that there is no HOLD on the job.

INDENT/NOINDENT

INDENT specifies that the job data should be shifted right the specified number of columns. The value can be from 0 to 65535.

NOINDENT explicitly specifies that there is to be no indention for the job.

LINECOUNT/NOLINECOUNT

LINECOUNT specifies the maximum number of lines that JES2 prints on each output page for the job. The number can be from 0 to 254. If specified as 0, JES2 does not eject to a new page when the number of output lines exceeds the page limit specified during JES2 initialization.

The alias for LINECOUNT is LINECT.

NOLINECOUNT explicitly specifies that no LINECOUNT is specified for the job.

Note: This job default command is for JES2 only.

PAGEDEF/NOPAGEDEF

PAGEDEF specifies the name of a library member used by Print Services Facility (PSF) for printing on a page mode printer. The name can be from 1 to 6 alphanumeric or national characters.

NOPAGEDEF explicitly specifies that no PAGEDEF is specified for the job.

PRMODE/NOPRMODE

PRMODE specifies the processing mode required to print the job.

- LINE specifies to schedule the job to a line mode printer.
- PAGE specifies to schedule the job to a page mode printer.

NOPRMODE explicitly specifies that no PRMODE is specified for the job.

TRC/NOTRC

TRC specifies to a 3800 printing subsystem that each logical record contains a Table Reference Character (TRC). The TRC identifies which character arrangement table in the CHARx parameters is to be used to print the line. Same as JCL parameter DD DCB=OPTCD=J.

NOTRC explicitly specifies that no TRC is specified for the job.

UCS/NOUCS

UCS specifies the Universal Character Set (UCS) to be used to print the job on a JES printer. The name can be from 1 to 4 alphanumeric or national characters.

NOUCS explicitly specifies that no UCS is specified for the job.

WIDTH/NOWIDTH

WIDTH specifies the width of the job data. It is the maximum line length before a new line is forced. The value can be from 0 to 65535.

NOWIDTH explicitly specifies that there is no WIDTH limitation for the job.

WRITER/NOWRITER

WRITER names an external writer. The name can be from 1 to 8 alphanumeric characters.

NOWRITER explicitly specifies that no WRITER is specified for the job.

Note: If you code DEST=*nodename* with the .userid extension concurrently with WRITER=*name* an MVS error message will result.

SET LOG

The **SET LOG** command changes parameter values associated with the log.

The alias for **SET LOG** is MODIFY LOG.

SET LOG [*PARAMETERS*]

Syntax Description

PARAMETERS Any parameters described in the following section, “[SET LOG Parameters](#)” on [page 1-52](#).

SET LOG Parameters

This section describes the valid parameters for the **SET LOG** command.

CLASS

The CLASS parameter specifies the JES system output (SYSOUT) class that SOLVE:EPS should use to spool the log. If the class is specified as *, the startup JCL message class (MSGCLASS) for SOLVE:EPS is used as the system output class.

This parameter takes effect at the next SPINLOG action.

SET LOG CLASS *class*

Syntax Description

class Specifies the system output class to be used.
The default for *class* is A.

DESTINATION/NODESTINATION

The DESTINATION parameter specifies the JES system output (SYSOUT) destination that SOLVE:EPS should use to spool the log. If this parameter is not specified the standard default JES destination is used.

This parameter takes effect at the next SPINLOG action.

SET LOG DESTINATION *dest*

Syntax Description

dest Specifies the desired system output destination.

FORMS/NOFORMS

The FORMS parameter specifies the JES system output (SYSOUT) form that SOLVE:EPS should use to spool the log. If this parameter is not specified the standard default JES form is used.

This parameter takes effect at the next SPINLOG action.

SET LOG FORMS *form*

Syntax Description

form Specifies the system output form to be used.
form can be from 1 to 4 alphanumeric or national characters.

LOGLEVEL

The LOGLEVEL parameter controls the types of messages that the log task itself writes to the log.

This parameter is dynamic.

SET LOG LOGLEVEL [FATAL | ERROR | WARNING] |
[INFORMATION | NOINFORMATION | DEBUG | NODEBUG]

Syntax Description

FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION NOINFORMATION	INFORMATION enables informational messaging (also sets NODEBUG). NOINFORMATION disables informational messaging (also sets NODEBUG).
DEBUG NODEBUG	DEBUG enables debug messaging (also sets INFORMATION). NODEBUG disables debug messaging (also sets INFORMATION).

MESSAGELEVEL

The MESSAGELEVEL parameter controls the types of messages that the log task itself writes to the MVS console via the WTO interface.

This parameter is dynamic.

The alias for MESSAGELEVEL is MSGLEVEL.

**SET LOG MESSAGELEVEL [FATAL | ERROR | WARNING] |
[INFORMATION | NOINFORMATION | DEBUG | NODEBUG]**

Syntax Description

FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION NOINFORMATION	INFORMATION enables informational messaging (also sets NODEBUG). NOINFORMATION disables informational messaging (also sets NODEBUG).
DEBUG NODEBUG	DEBUG enables debug messaging (also sets INFORMATION). NODEBUG disables debug messaging (also sets INFORMATION).

SPINRECORDS

The SPINRECORDS parameter specifies the number of records to be written to the log before it is closed and reopened (log spinning—an internal **SPIN** command is issued). The **SPIN** command is automatically issued after the specified number of records.

If this parameter is non-zero, log spinning occurs based on record counts even if SPINTIME is specified. When the spin occurs, the currently specified system output (SYSOUT) parameters will be used to allocate the new log.

This parameter takes effect immediately and could cause a SPIN LOG action.

The default for the SPINRECORDS parameter is 0.

SET LOG SPINRECORDS *recs*

Syntax Description

<i>recs</i>	Number of records to be written to the log. If specified as 0, the log will not be closed and reopened based on records. The range is 1000 to 1,000,000,000.
-------------	--

SPINSYNCHRONIZE/NOSPINSYNCHRONIZE

The SPINSYNCHRONIZE parameter indicates, when SPINTIME is in effect, that the log close and reopen (log spinning; an internal **SPIN** command is issued) operation will occur synchronized with the time of day (in other words, log close/reopen occurs on the hour, daily log close/reopen occurs at midnight, etc.).

This parameter is dynamic. It takes effect at the next SPINLOG action.

SET LOG SPINSYNCHRONIZE

The NOSPINSYNCHRONIZE parameter indicates, when SPINTIME is in effect, that the log close and reopen (spinning) operation will occur as measured from the start of logging or last spin.

SET LOG NOSPINSYNCHRONIZE**SPINTIME**

The SPINTIME parameter specifies the time interval in minutes before the log will be closed and reopened (log spinning; an internal **SPIN** command is issued). If this parameter is specified, log spinning will occur at the specified time interval even if SPINRECS has just been recently reached. When SPINTIME occurs, the count of the number of records will be reset to zero. When the spin occurs, the currently specified system output (SYSOUT) parameters will be used to allocate the new log.

This parameter takes effect at the next SPINLOG action.

The default is no spinning will occur based on time interval.

SET LOG SPINTIME *time***Syntax Description**

<i>time</i>	Specifies in minutes the amount of time to elapse before the log will be closed and reopened. The range is from 1 to 50400 (five weeks).
-------------	--

CAUTION ! If SPINTIME and SPINRECORDS are specified together, spinning may occur very frequently. Monitor the SOLVE:EPS job to see if modifications to these parameters may be required to avoid excessive log spinning.

UPPERCASE/NOUPPERCASE

The UPPERCASE parameter specifies that messages will be written to the SOLVE:EPS log in uppercase. Messages are normally written in upper and lower (mixed) case. The UPPERCASE parameter causes the entire message to be written in upper case.

This parameter is dynamic. It takes effect on the next log record written.

The default is NOUPPERCASE.

Note: Some of the parameters in SOLVE:EPS are specified in lower case and writing messages that contain these parameters to the log in upper case may cause confusion.

SET LOG UPPERCASE

The NOUPPERCASE parameter sets the writing of messages in the log to mixed case.

SET LOG NOUPPERCASE

SET OUTPROCESSOR

The **SET OUTPROCESSOR** command changes parameter values associated with an Outprocessor.

These parameters are also valid on a **DEFINE OUTPROCESSOR** command.

The alias for **SET OUTPROCESSOR** is **MODIFY OUTPROCESSOR**.

SET OUTPROCESSOR *outp_name* [*PARAMETERS*]

Syntax Description

outp_name Name of the Outprocessor to be set. The Outprocessor must have been previously defined with a **DEFINE OUTPROCESSOR** command.

PARAMETERS Any of the parameters described in the following sections, [“SET OUTPROCESSOR Parameters” on page 1-56](#), [“BSD-Specific Parameters” on page 1-60](#), and [“NJE-Specific Parameters” on page 1-61](#).

Note: Not all of the parameters are applicable to all types of Outprocessors. If a parameter is specified and is not applicable to the type of Outprocessor, an error message will be issued.

SET OUTPROCESSOR Parameters

This section describes the parameters for the **SET OUTPROCESSOR** command.

CONVERTSELECTOR

The CONVERTSELECTOR user exit provides flexibility in converter selection. It is called by converter initialization after a converter has been selected. It may override the selection, or it may abort the conversion process.

The user exit is compiled and loaded when the first Outprocessor referencing it is started. If it contains errors, the Outprocessor will not start.

The argument to this parameter specifies an EPSUSEX user exit:

The CONVERTSELECTOR parameter is not dynamic.

SET OUTPROCESSOR *outp_name* **CONVERTSELECTOR** *mem_name*

Syntax Description

outp_name Specifies the name of the Outprocessor to be set.

mem_name Specifies the name of the EPSUSEX user exit in the PARMLIB DD data set.

To remove the convert selector user exit from an Outprocessor, use the **NOCONVERTSELECTOR** parameter.

SET OUTPROCESSOR *outp_name* **NOCONVERTSELECTOR**

DEFAULT/NODEFAULT

The **DEFAULT** parameter makes an Outprocessor model the default for subsequent **DEFINE OUTPROCESSOR** commands.

A **LIKE** parameter need not be specified on subsequent **DEFINE OUTPROCESSOR** commands to have the new definition take on the same parameters as the **DEFAULT** definition. If a default has already been set, this replaces that setting.

This parameter is dynamic.

SET OUTPROCESSOR *outp_name* **DEFAULT**

Syntax Description

<i>outp_name</i>	Name of an Outprocessor model entry. If the named Outprocessor is not a model, an error message results.
------------------	--

NODEFAULT removes the **DEFAULT** attribute from an Outprocessor model. Subsequent **DEFINE OUTPROCESSOR** commands will have no default model to alter the internal Outprocessor defaults.

SET OUTPROCESSOR *outp_name* **NODEFAULT**

ENVIRONMENTMEMBER

The **ENVIRONMENTMEMBER** parameter lets you set C runtime environment variables for BSD Inprocessors and Outprocessors. They are specified by assigning a PARMLIB DD member to the Inprocessors and Outprocessors. The environment settings are specified in the member.

SET OUTPROCESSOR *outp_name* **ENVIRONMENTMEMBER** *mem_name*

Syntax Description

<i>outp_name</i>	Name of a BSD Outprocessor.
<i>mem_name</i>	Name of the PARMLIB DD member containing environment variable settings.

LOGLEVEL

The LOGLEVEL parameter controls the types of messages written to the log for the Outprocessor.

This parameter is dynamic.

SET OUTPROCESSOR *outp_name* **LOGLEVEL** [**FATAL** | **ERROR** | **WARNING**] |
[**INFORMATION** | **NOINFORMATION** | **DEBUG** | **NODEBUG**]

Syntax Description

<i>outp_name</i>	Name of the Outprocessor being set.
FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION NOINFORMATION	INFORMATION enables informational messaging (also sets NODEBUG). NOINFORMATION disables informational messaging (also sets NODEBUG).
DEBUG NODEBUG	DEBUG enables debug messaging (also sets INFORMATION). NODEBUG disables debug messaging (also sets INFORMATION).

MESSAGELEVEL

The MESSAGELEVEL parameter controls the types of messages written to the MVS console via WTO for the Outprocessor.

This parameter is dynamic.

The alias for MESSAGELEVEL is MSGLEVEL.

SET OUTPROCESSOR *outp_name* **MESSAGELEVEL** [**FATAL** | **ERROR** | **WARNING**] |
[**INFORMATION** | **NOINFORMATION** | **DEBUG** | **NODEBUG**]

Syntax Description

<i>outp_name</i>	Name of the Outprocessor being set.
FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION NOINFORMATION	INFORMATION enables informational messaging (also sets NODEBUG). NOINFORMATION disables informational messaging (also sets NODEBUG).
DEBUG NODEBUG	DEBUG enables debug messaging (also sets INFORMATION). NODEBUG disables debug messaging (also sets INFORMATION).

SESSIONS

The SESSIONS parameter specifies the number of jobs the Outprocessor can process simultaneously.

This parameter varies according to type. Currently, an NJE Outprocessor can have only one session; the JES and BSD Outprocessors can have any number desired, depending on memory and CPU availability. The default is 1.

This parameter is dynamic.

SET OUTPROCESSOR *outp_name* **SESSIONS** *sess_num*

Syntax Description

<i>outp_name</i>	Name of the Outprocessor to be set.
<i>sess_num</i>	The number of sessions. This value must be numeric. -1 signifies unlimited.

TRACEWORD

The TRACEWORD parameter specifies trace settings for the Outprocessor.

This parameter is dynamic.

SET OUTPROCESSOR *inp_name* **TRACEWORD** *trace_value*

Syntax Description

<i>inp_name</i>	Name of the Outprocessor to be set.
<i>trace_value</i>	Value of the TRACEWORD to be set.

Read [“Trace Control” on page 2-9](#) in the chapter [“SOLVE:EPS Operation”](#) for the valid TRACEWORD values.

BSD-Specific Parameters

This section describes the valid BSD-specific parameters for the **SET OUTPROCESSOR** command.

SYSID

The SYSID parameter defines, for a BSD Outprocessor, the MVS subsystem name of the SOLVE:TCPaccess address space. The SYSID parameter is used only by the BSD Outprocessor.

This parameter is not dynamic.

SET OUTPROCESSOR *outp_name* **SYSID** *subsysid*

Syntax Description

<i>outp_name</i>	Name of the Outprocessor to be set.
<i>subsysid</i>	Defines the MVS subsystem name. The default is ACSS.

DSO-Specific Parameters

This section describes the valid DSO-specific parameters for the SET OUTPROCESSOR command.

SYSID

The SYSID parameter defines, for a DSO Outprocessor, the MVS subsystem name of the SOLVE:TCPaccess address space. The SYSID parameter is used only by the DSO Outprocessor.

This parameter is not dynamic.

SET OUTPROCESSOR *outp_name* **SYSID** *subsys_id*

Syntax Description

<i>outp_name</i>	Name of the Outprocessor to be set.
<i>subsys_id</i>	Defines the MVS subsystem name. The default is ACSS.

NJE-Specific Parameters

The NJE parameters define an NJE interface with JES. An NJE interface has both an input and an output function. Within SOLVE:EPS the Inprocessor is used to define all of the parameters that connect the SOLVE:EPS NJE interface with the JES NJE interface.

Since these parameters need to be defined only once, the Outprocessor definition only needs to be related to the Inprocessor definition. This is done with the NJEINPROC parameter.

NJEINPROCESSOR

The NJEINPROCESSOR parameter specifies the name of the NJE Inprocessor to which this NJE Outprocessor is associated. The Inprocessor and the Outprocessor objects for NJE are closely coupled due to the design of the NJE interface by JES. Therefore, by setting up the Inprocessor with all its NJE-specific parameters, the Outprocessor needs to inherit those same NJE-specific parameters. This parameter is checked when the Outprocessor is started to ensure that the Inprocessor is defined and an error will occur at **START** command time.

This parameter is not dynamic.

SET OUTPROCESSOR *outp_name* **NJEINPROCESSOR** *inp_name*

Syntax Description

<i>outp_name</i>	Name of the Outprocessor to be set.
<i>inp_name</i>	Name of the Inprocessor to be specified. If the named Inprocessor is not defined, <i>outp_name</i> will default to the same name as the Outprocessor name. Hence, when the Outprocessor is started it will match up to an Inprocessor of the same name.

SET QUEUE

The **SET QUEUE** command changes parameter values associated with a queue.

These parameters are also valid on a **DEFINE QUEUE** command.

The alias for **SET QUEUE** is **MODIFY QUEUE**.

SET QUEUE *q_name* [*PARAMETERS*]

Syntax Description

<i>q_name</i>	Name of the queue to be set. The queue must have been defined with a DEFINE QUEUE command.
<i>PARAMETERS</i>	Any parameters described in the following sections, “SET QUEUE Parameters” on page 1-62 and “BSD-Specific Parameters” on page 1-70 .

Note: Some of the parameters can be set dynamically and others must be set when the queue is stopped. This behavior is noted in each parameter definition.

SET QUEUE Parameters

This section describes the valid parameters for the **SET QUEUE** command.

DATASETEXIT/NODATASETEXIT

The DATASETEXIT parameter sets the name of a Dataset User Exit to associate with the queue. If the exit is written in Assembler, an executable file must exist in the SOLVE:EPS load library concatenation; if the exit is written in EPSUSEX, the source code must be in the PARMLIB DD for SOLVE:EPS.

For an EPSUSEX data set exit, the code is loaded and parsed when the first queue referencing it is started.

This parameter is not dynamic.

The default value is NODATASETEXIT.

SET QUEUE *q_name* **DATASETEXIT MODULE** *mod_name*

or

SET QUEUE *q_name* **DATASETEXIT MEMBER** *mem_name*

Syntax Description

<i>q_name</i>	Specifies the name of the queue to be set.
<i>mod_name</i>	Specifies the name of an executable Dataset User Exit module in the SOLVE:EPS load library concatenation.
<i>mem_name</i>	Specifies the member name of an EPSUSEX Dataset User Exit in the PARMLIB DD.

NODATASETEXIT specifies to associate no Dataset User Exit with the queue.

DEFAULT/NODEFAULT

The DEFAULT parameter makes a queue model the default for subsequent **DEFINE QUEUE** commands.

A LIKE parameter need not be specified on subsequent **DEFINE QUEUE** commands to have the new definition take on the same parameters as the DEFAULT definition. If a default has already been set, this replaces that setting.

This parameter is dynamic.

SET QUEUE *q_model_name* DEFAULT

Syntax Description

q_model_name Name of a queue model entry. If the named queue is not a model an error message results.

NODEFAULT removes the DEFAULT attribute from a queue model. Subsequent **DEFINE QUEUE** commands will have no default model to alter the internal queue defaults.

DESCRIPTION

The DESCRIPTION parameter associates an arbitrary value with a queue. It could be used for such things as describing the type of printer that the queue represents.

This parameter is dynamic.

SET QUEUE *q_name* DESCRIPTION *desc*

Syntax Description

q_name Name of the queue to be set

desc Description to assign to the queue. It can be a maximum of 32 characters long; blank characters are not allowed.

DISPATCHEXIT/NODISPATCHEXIT

The DISPATCHEXIT parameter sets the name of a Dispatch User Exit to associate with the queue. The source code for the exit must exist in the PARMLIB DD for SOLVE:EPS under the given member name.

The exit is loaded and parsed when the first queue referencing it is started.

This parameter is not dynamic.

The default value is NODISPATCHEXIT.

SET QUEUE *q_name* DISPATCHEXIT *mem_name*

Syntax Description

q_name Name of the queue to which the exit is being assigned.

mem_name Member name of the Dispatch User Exit in the PARMLIB DD containing the EPSUSEX code for the exit.

DISPOSITION/NODISPOSITION

The DISPOSITION parameter specifies the disposition of the job after it has been successfully sent via an Outprocessor.

This parameter is not dynamic.

The default DISPOSITION parameter on the **SET QUEUE** command is NODISPOSITION.

The default option for the DISPOSITION parameter is PURGE.

SET QUEUE *q_name* DISPOSITION PURGE | HOLD

Syntax Description

<i>q_name</i>	Name of the queue to be set.
PURGE	Job will be purged when normal outprocessing is complete. The alias for PURGE is DELETE.
HOLD	Job will be held in the SOLVE:EPS spool when normal outprocessing is complete.

LINESIZE

The LINESIZE parameter specifies the maximum line size for LU1 or LU3 LU names. This applies only to the VTAM virtual printers when input is via VTAM.

This parameter is not dynamic.

SET QUEUE *q_name* LINESIZE *num_chars*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>num_chars</i>	Line size; a new line will occur every <i>num_chars</i> characters if no newline has been detected. <i>num_chars</i> is any number from 0 to 255. For LU1, zero sets the line size to infinite; for LU3, zero defaults to the maximum value of 255. Default line size is 255.

LOGLEVEL

The LOGLEVEL parameter controls the types of messages written to the log for the queue.

This parameter is dynamic.

SET QUEUE *q_name* LOGLEVEL [FATAL | ERROR | WARNING] |
[INFORMATION | NOINFORMATION | DEBUG | NODEBUG]

Syntax Description

<i>q_name</i>	Name of the queue to be set.
FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION NOINFORMATION	INFORMATION enables informational messaging (also sets NODEBUG). NOINFORMATION disables informational messaging (also sets NODEBUG).
DEBUG NODEBUG	DEBUG enables debug messaging (also sets INFORMATION). NODEBUG disables debug messaging (also sets INFORMATION).

MESSAGELEVEL

The MESSAGELEVEL parameter controls the types of messages written to the MVS console via WTO for the named queue.

This parameter is dynamic.

The alias for MESSAGELEVEL is MSGLEVEL.

**SET QUEUE *q_name* MESSAGELEVEL [FATAL | ERROR | WARNING] |
[INFORMATION | NOINFORMATION | DEBUG | NODEBUG]**

Syntax Description

<i>q_name</i>	Name of the queue to be set.
FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION NOINFORMATION	INFORMATION enables informational messaging (also sets NODEBUG). NOINFORMATION disables informational messaging (also sets NODEBUG).
DEBUG NODEBUG	DEBUG enables debug messaging (also sets INFORMATION). NODEBUG disables debug messaging (also sets INFORMATION).

OUTPROCESSOR

The OUTPROCESSOR parameter associates a queue with a particular Outprocessor. This parameter is required before a QUEUE can be started and must not reference an Outprocessor model.

This parameter is not dynamic and has no default.

SET QUEUE *q_name* OUTPROCESSOR *outp_name*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>outp_name</i>	Name of the Outprocessor that processes jobs on this queue.

PAGESIZE

The PAGESIZE parameter specifies the maximum page size for LU1 LU names. This applies only to the VTAM virtual printers when input is via VTAM.

This parameter is not dynamic.

SET QUEUE *q_name* PAGESIZE *num_lines*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>num_lines</i>	Page size; a new page will occur every <i>num_lines</i> if no new page has been detected. <i>num_lines</i> is any number from 0 to 255; a value of zero sets the page size to infinite. Default page size is 255 lines.

RETRYCOUNT

The RETRYCOUNT parameter specifies the number of consecutive allowable retry attempts before the job fails. If this number is exceeded, the queue will stop itself. The number range is 1 to 32767.

This parameter is dynamic.

The alias for RETRYCOUNT is RCNT.

SET QUEUE *q_name* RETRYCOUNT *count*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>count</i>	Number of times the queue can be retried before it is autostopped. The default value is 5.

RETRYINTERVAL

The RETRYINTERVAL parameter specifies an interval, in seconds, for SOLVE:EPS to use before retrying the queue to dispatch jobs on the Outprocessor.

If the queue is currently waiting for a retry interval, setting this parameter will not reset the current waiting interval the queue is using.

This parameter is dynamic and takes effect at the next retry cycle.

The default for the RETRYINTERVAL parameter is 300 (5 minutes).

The alias for RETRYINTERVAL is RINT.

SET QUEUE *q_name* RETRYINTERVAL *seconds*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>seconds</i>	Number of seconds the queue will wait before dispatching jobs again on the Outprocessor. The range is 1 to 2,147,483,647.

SENDALLCOPIES

SENDALLCOPIES determines how multiple-copy files are handled by BSD Outprocessors.

SET QUEUE *q_name* SENDALLCOPIES

SET QUEUE *q_name* NOSENDALLCOPIES

Syntax Description

q_name Name of the Queue to modify.

The default setting is NOSENDALLCOPIES, which causes one copy of a multiple-copy file to be sent to the remote BSD system with multiple print commands in the control file.

SESSIONS

The SESSIONS parameter specifies the number of jobs on this queue that can be processed by the Outprocessor at any one time. This parameter is typically used for direct network-attached printers that can accept only one job at a time.

This parameter is dynamic.

SET QUEUE *q_name* SESSIONS *sess_num*

Syntax Description

q_name Name of the queue to be set.

sess_num Number of sessions. A value of -1 specifies no session limit on the queue. The Outprocessor controls the number of sessions.
The default is -1.

TRANSLATE

TRANSLATE indicates the desired language code set and encoding supported by the printer this queue represents. Translation changes the encoding of a SOLVE:EPS job from EBCDIC to ASCII and from ASCII to EBCDIC. If NONE is specified, the data is not translated.

Refer to the translate member of the SOLVE:EPS PARMLIB DD for supported translations. This member also specifies the default translation.

The default is UNSPECIFIED.

This parameter is not dynamic.

The alias for TRANSLATE is XLATE.

SET QUEUE *q_name* TRANSLATE UNSPECIFIED | *lang_name* | NONE

Syntax Description

<i>q_name</i>	Name of the queue to be set.
UNSPECIFIED	There is no translation specified for this queue. Translation is determined by the Inprocessor and the translate member of the PARMLIB DD. This is the default. The alias for UNSPECIFIED is UNSET.
<i>lang_name</i>	Name of the desired language translation. This value must exist in the translate member of the PARMLIB DD.
NONE	No translation be used for the queue.

WIDTH/NOWIDTH

WIDTH specifies the width of the job data. It is the maximum line length before a new line is forced. The value can be from 0 to 65535.

NOWIDTH explicitly specifies that there is no WIDTH limitation for the job.

BSD-Specific Parameters

This section describes the valid BSD-specific parameters for the **SET QUEUE** command.

BANNER/NOBANNER

The BANNER parameter turns on banner pages for printing at the remote printer. This is done by sending the L control file option to the remote host.

This parameter is dynamic.

SET QUEUE *q_name* BANNER

Syntax Description

q_name Name of the queue to be set.

NOBANNER turns off banner pages from printing at the remote printer.

This is done by not sending the L control file option to the remote host. If this parameter is not specified, banner pages will be printed if that is the default on the remote host.

SET QUEUE *q_name* NOBANNER

CONTROLFILE

The CONTROLFILE parameter specifies the position of the control file for a BSD Outprocessor. The value can be either FIRST or LAST; the default is LAST.

This parameter is not dynamic.

The default is LAST.

The alias for CONTROLFILE is CFILE or CTLFILE.

SET QUEUE *q_name* CONTROLFILE FIRST | LAST

Syntax Description

q_name Name of the queue to be set.

FIRST Specifies that the control file be sent to the remote host before data.

LAST Specifies that the control file be sent to the remote host after data.

Note: Some PC LPD server implementations require that the control file be sent first.

CONTROLLINEEXIT/NOCONTROLLINEEXIT

The CONTROLLINEEXIT parameter specifies a module to be called by a BSD Outprocessor as it builds control file lines to send to a remote host. This is User Exit 2. Read the ["User Exits"](#) appendix for details.

This parameter is not dynamic.

The alias for CONTROLLINEEXIT is CTLLINEEXIT.

SET QUEUE *q_name* CONTROLLINEEXIT *mod_name*

SET QUEUE *q_name* NOCONTROLLINEEXIT

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>mod_name</i>	Name of an executable module in the SOLVE:EPS JOBLIB or STEPLIB concatenation.

The default value is no user exit.

NOCONTROLLINEEXIT resets the queue to its default value.

CONTROLSTRINGPREFIX/NOCONTROLSTRINGPREFIX

The CONTROLSTRINGPREFIX parameter specifies printer control codes that are placed before the print data when a job on the specified queue is processed by a BSD Outprocessor.

This parameter is not dynamic.

The alias for the CONTROLSTRINGPREFIX parameter is CTLSTRPRE.

SET QUEUE *q_name* CONTROLSTRINGPREFIX *prefix* | MEMBER *mem_name*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>prefix</i>	Prefix data that will be placed before the print data. The prefix must be hexadecimal data and can be a maximum of 64 characters; this data will not be translated.
<i>mem_name</i>	Name of a member in T07PARM.

Read the section "Control Strings" in "SOLVE:EPS Members" (Chapter 8) of the ***SOLVE:EPS Installation and Administration Guide*** for more information.

SET QUEUE *q_name* CONTROLSTRINGPREFIX 1B266C314F

SET QUEUE *q_name* CONTROLSTRINGPREFIX MEMBER HPPRE

CONTROLSTRINGSUFFIX/NOCONTROLSTRINGSUFFIX

The CONTROLSTRINGSUFFIX parameter specifies printer control codes that are placed after the print data when a job on the specified queue is processed by a BSD Outprocessor.

This parameter is not dynamic.

The alias for the CONTROLSTRINGSUFFIX parameter is CTLSTRSUF.

SET QUEUE *q_name* CONTROLSTRINGSUFFIX *suffix* | MEMBER *mem_name*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>suffix</i>	Suffix data that will be placed after the print data. The suffix must be hexadecimal data and can be a maximum of 64 characters; this data will not be translated.
<i>mem_name</i>	Name of a member in T07PARM.

Read the section “Control Strings” in “SOLVE:EPS Members” (Chapter 8) of the ***SOLVE:EPS Installation and Administration Guide*** for more information.

SET QUEUE *q_name* CONTROLSTRINGSUFFIX 1645

SET QUEUE *q_name* CONTROLSTRINGSUFFIX MEMBER HPSUF

CONVERSIONCACHESIZE

Since a BSD Outprocessor processes SOLVE:EPS temporary data sets twice (once to get the size and once to transmit to the remote destination), an already heavily loaded system can have problems. If you cache the results of the conversion during file sizing, you can avoid a second pass of the data during file transfer. Use the CONVERSIONCACHESIZE parameter to customize the cache size for a BSD queue.

This parameter is dynamic.

The alias for CONVERSIONCACHESIZE is CNVCACHESIZE.

SET QUEUE *q_name* CONVERSIONCACHESIZE *size*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>size</i>	Size to set the conversion cache to. Valid values for <i>size</i> are: DEFAULT - Use the default cache size 0 - Same as DEFAULT <i>number</i> - A positive decimal number greater than or equal to 1024 (one kilobyte). Suffixes K (kilobytes) and M (megabytes) are recognized.

FILENAMETIMESTAMP

FILENAMETIMESTAMP controls the suffix of a date and time stamp to network file names for remote BSD systems.

SET QUEUE *q_name* FILENAMETIMESTAMP

SET QUEUE *q_name* NOFILENAMETIMESTAMP

Syntax Description

q_name Name of the Queue to modify.

The default setting is FILENAMETIMESTAMP.

INDENT/NOINDENT

INDENT specifies that the job data should be shifted right the specified number of columns. The value can be from 0 to 65535.

SET QUEUE *q_name* INDENT *num*

NOINDENT explicitly specifies that there is to be no indention for the job.

FILTER

The FILTER parameter specifies the type of LPD file filter to use.

This parameter is dynamic.

The default is FILTER f.

SET QUEUE *q_name* FILTER *filter*

Syntax Description

q_name Name of the queue to be set

filter One of these filters:

- c—Caltech Intermediate Form (CIF) file will be plotted.
- d—DVI (TeX output) file will be printed.
- f—Plain text file will be printed.
- g—Berkeley UNIX file will be plotted.
- l—File will be printed without filtering ASCII control characters.
- n—ditroff file will be printed.
- o—Postscript file will be printed.
- p—pr-form file will be printed.
- r—FORTRAN carriage control file will be printed.
- t—troff file will be printed.
- v—Sun raster file will be printed.

LOCALDELAYPORTTIMER

The LOCALDELAYPORTTIMER parameter specifies the amount of time to delay making local port numbers available after a SOLVE:EPS job completes on a BSD-type queue. Port numbers for a BSD queue are determined by the **SET QUEUE LOCALPORTS** command.

This parameter is dynamic.

The default LOCALDELAYPORTTIMER value is 5 seconds.

The alias for LOCALDELAYPORTTIMER is LDELAYPORTTIMER.

SET QUEUE *q_name* LOCALDELAYPORTTIMER *value*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>value</i>	Delay time in seconds. The range is from 1 to 600.

LOCALHOSTNAME/NOLOCALHOSTNAME

The LOCALHOSTNAME parameter specifies the host name used by a BSD Outprocessor in the LPR protocol H control record. By default, the system is queried for a local host name. This parameter allows the user to override this default and is included for compatibility with Version 2.0.

This parameter is not dynamic.

The alias for LOCALHOSTNAME is HOSTNAME.

SET QUEUE *q_name* LOCALHOSTNAME *host_name*

SET QUEUE *q_name* NOLOCALHOSTNAME *host_name*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>host_name</i>	name by which SOLVE:EPS will identify itself to the remote host. The default value is no override.

NOLOCALHOSTNAME resets the queue to its default value.

LOCALPORTS

The LOCALPORTS parameter specifies the range of local ports that a BSD Outprocessor will use for this queue.

The default is the range 721 through 731. The LOCALPORTS range will affect the number of sessions that can be achieved by this queue. With the defaults, the maximum number of sessions for the queue is 11. Therefore, specifying a SESSIONS parameter on a queue of greater than 11 has no effect.

Note: Be aware that another SOLVE:EPS using the same TCP address space, or other printing products using the LPR protocol, may cause conflicts in LOCALPORTS usage. If this occurs, there may be delays in printing from SOLVE:EPS due to “port in use” conflicts. Also, other queues using the same REMOTEHOST and REMOTEPORT may have conflicts with LOCALPORTS internally within SOLVE:EPS which will cause Outprocessor concurrency to be less than expected.

This parameter is not dynamic.

The alias for LOCALPORTS is LPORTS.

SET QUEUE *q_name* **LOCALPORTS** *port_num1* *port_num2*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>port_num1</i>	Beginning port number of the local ports range to be used. The default is 721.
<i>port_num2</i>	Ending port number of the local ports range to be used. The default is 731.

REMOTEHOST

The REMOTEHOST parameter specifies the fully-qualified host name, partially-qualified host name, or IP address of a remote host for this queue object.

This parameter is required before a BSD QUEUE can be started. There is no default.

This parameter is not dynamic.

The alias for REMOTEHOST is RHOST.

SET QUEUE *q_name* **REMOTEHOST** *r_host_name* | *nnn.nnn.nnn.nnn*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>r_host_name</i>	Fully-qualified host name or partially-qualified host name. The name is in Internet notation of <i>name.name.name</i> . A <code>gethostbyname</code> lookup will be performed to resolve the <i>r_host_name</i> to its actual IP address.
<i>nnn.nnn.nnn.nnn</i>	IP address of the remote host. <i>nnn</i> is a decimal number from 0 to 255. This is the normal dotted-decimal Internet format for IP addresses. All numbers supplied as parts in a dot notation can be decimal, octal, or hexadecimal. A leading 0x or 0X implies hexadecimal; otherwise, a leading 0 implies octal; if there is no leading 0, the number is interpreted as decimal.

REMOTEPORT

The REMOTEPORT parameter specifies the remote port for the BSD LPD server on the remote host.

This parameter is not dynamic.

The default is 515.

The alias for REMOTEPORT is RPORT.

SET QUEUE *q_name* **REMOTEPORT** *port_num*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>port_num</i>	Remote host server port number.

REMOTEPRINTER

The REMOTEPRINTER parameter specifies the queue on the remote host for BSD LPD printing. The value is case sensitive for UNIX systems that require mixed case.

This parameter is not dynamic.

Aliases for REMOTEPRINTER are RQUEUE and RPRINTER.

SET QUEUE *q_name* **REMOTEPRINTER** *r_printer*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>r_printer</i>	Defines the printer queue on the remote host as defined in the /etc/printcap file or its equivalent. If <i>r_printer</i> is omitted, the default is lp.

DSO-Specific Parameters

This section describes the valid DSO-specific parameters for the SET QUEUE command.

CONTROLSTRINGPREFIX/NOCONTROLSTRINGPREFIX

The CONTROLSTRINGPREFIX parameter specifies printer control codes that are placed before the print data when a job on the specified queue is processed by a DSO Outprocessor.

This parameter is not dynamic.

The alias for the CONTROLSTRINGPREFIX parameter is CTLSTRPRE.

SET QUEUE *q_name* CONTROLSTRINGPREFIX *prefix* | MEMBER *mem_name*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>prefix</i>	Prefix data that will be placed before the print data. The prefix must be hexadecimal data and can be a maximum of 64 characters; this data will not be translated.
<i>mem_name</i>	Name of a member in T07PARM.

Read the section “Control Strings” in “SOLVE:EPS Members” (Chapter 8) of the **SOLVE:EPS Installation and Administration Guide** for more information.

SET QUEUE *q_name* CONTROLSTRINGPREFIX 1B266C314F

SET QUEUE *q_name* CONTROLSTRINGPREFIX MEMBER HPPRE

CONTROLSTRINGSUFFIX/NOCONTROLSTRINGSUFFIX

The CONTROLSTRINGSUFFIX parameter specifies printer control codes that are placed after the print data when a job on the specified queue is processed by a DSO Outprocessor.

This parameter is not dynamic.

The alias for the CONTROLSTRINGSUFFIX parameter is CTLSTRSUF.

SET QUEUE *q_name* CONTROLSTRINGSUFFIX *suffix* | MEMBER *mem_name*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>suffix</i>	Suffix data that will be placed after the print data. The suffix must be hexadecimal data and can be a maximum of 64 characters; this data will not be translated.
<i>mem_name</i>	Name of a member in T07PARM.

Read the section “Control Strings” in “SOLVE:EPS Members” (Chapter 8) of the **SOLVE:EPS Installation and Administration Guide** for more information.

SET QUEUE *q_name* CONTROLSTRINGSUFFIX 1645

SET QUEUE *q_name* CONTROLSTRINGSUFFIX MEMBER HPSUF

REMOTEHOST

The REMOTEHOST parameter specifies the fully-qualified host name, partially-qualified host name, or IP address of a remote host for this queue object.

This parameter is required before a DSO QUEUE can be started. There is no default.

This parameter is not dynamic.

The alias for REMOTEHOST is RHOST.

SET QUEUE *q_name* **REMOTEHOST** *rhost_name* | *nnn.nnn.nnn.nnn*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>rhost_name</i>	Fully-qualified host name or partially-qualified host name. The name is in Internet notation of <i>name.name.name</i> . A gethostbyname lookup will be performed to resolve the <i>r_host_name</i> to its actual IP address.
<i>nnn.nnn.nnn.nnn</i>	IP address of the remote host. <i>nnn</i> is a decimal number from 0 to 255. This is the normal dotted-decimal Internet format for IP addresses. All numbers supplied as parts in a dot notation can be decimal, octal, or hexadecimal. A leading 0x or 0X implies hexadecimal; otherwise, a leading 0 implies octal. If there is no leading 0, the number is interpreted as decimal.

REMOTEPORT

The REMOTEPORT parameter specifies the remote port for the DSO LPD server on the remote host.

This parameter is not dynamic.

This parameter defaults to 515; however, in order to print to a LAN-connected printer you must override this value to 9100.

The alias for REMOTEPORT is RPORT.

SET QUEUE *q_name* **REMOTEPORT** *port_num*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>port_num</i>	Remote host server port number.

CONVERSIONCACHESIZE

The CONVERSIONCACHESIZE parameter specifies the buffer size that the DSO Outprocessor is to use to transmit job output to remote printers.

This parameter is dynamic.

The alias for CONVERSIONCACHESIZE is CNVCACHESIZE.

SET QUEUE *q_name* **CONVERSIONCACHESIZE** *size*

Syntax Description

<i>q_name</i>	Name of the queue to be set.
<i>size</i>	Size to set the transmission buffer size. Valid values for size are: • DEFAULT—Use the default buffer size (4K) • 0—Same as DEFAULT • <i>number</i>—A positive decimal number greater than or equal to 1024 (one kilobyte). Suffixes K (kilobytes) and M (megabytes) are recognized.

SET SPOOL

The **SET SPOOL** command changes system-wide defaults for the spool (data files) that SOLVE:EPS creates. These spool files, or temporary files, contain the data for SOLVE:EPS jobs.

These parameters can be changed at any time and take effect the next time an Inprocessor needs to create a spool file for a job.

The alias for **SET SPOOL** is **MODIFY SPOOL**.

SET SPOOL [PARAMETERS]

Syntax Description

PARAMETERS Any parameters described in the following section, [“SET SPOOL Parameters” on page 1-81](#).

SET SPOOL Parameters

This section describes the valid parameters for the **SET SPOOL** command.

ALLOCATION

The **ALLOCATION** parameter specifies the type of allocation SOLVE:EPS will use for the data files.

There is no default. However, if this parameter is not specified, allocation will occur either by blocks or by average logical record if **AVGREC** is specified. The blocksize or average logical record used by SOLVE:EPS is 24,376 bytes. The **PRIMARYSPACE** and **SECONDARYSPACE** parameters then correspond to the number of blocks or the number of records.

This parameter is not dynamic.

SET SPOOL ALLOCATION TRACK | CYLINDER

Syntax Description

- | | |
|----------|-------------------------------------|
| TRACK | Allocate the data set in tracks. |
| CYLINDER | Allocate the data set in cylinders. |

AVERAGERECORDS

The AVERAGERECORDS parameter specifies the quantity that the primary and secondary space parameter values represent. This parameter can only be used if the spool data sets are managed by Storage Management System (SMS).

The alias for AVERAGERECORDS is AVGREC.

SET SPOOL AVERAGERECORDS U | K | M

Syntax Description

U	Allocation is in units.
K	Allocation is in thousands.
M	Allocation is in millions.

Read the *MVS/ESA SP V5 JCL Reference* and the *MVS/ESA SP V5 JCL User's Guide* for more information.

BLOCKSIZE

The BLOCKSIZE parameter allows for the specification of a block size to be used by SOLVE:EPS when creating spool/temp files. In the past the value was defaulted to 23476 and was unchangeable. Now the default value is 27998 and can be modified dynamically via the command. The entered value will be used for the next creation of a spool/temp file. The range of the value is 20-32760.

SET SPOOL BLOCKSIZE *block*

DATACLASS

The DATACLASS parameter specifies the DFSMS data class name that SOLVE:EPS will use for allocation of the data files.

There is no default.

SET SPOOL DATACLASS *data_clas*

Syntax Description

<i>data_clas</i>	DFSMS data class used to allocate data files.
------------------	---

Read the *MVS/ESA SP V5 JCL Reference* and the *MVS/ESA SP V5 JCL User's Guide* for more information.

HIGHLEVELQUALIFIER

The HIGHLEVELQUALIFIER parameter specifies the high level index for the data set names that will use for the data files. Other information will be appended to the SPOOL INDEX parameter to make a unique data set name. The information that is appended to the HIGHLEVELQUALIFIER is SOLVE:EPS job ID and data set sequence number. The maximum length is 35.

The default is EPS.SPOOL.

The alias for HIGHLEVELQUALIFIER is HLQ

SET SPOOL HIGHLEVELQUALIFIER *dsn_level*

Syntax Description

dsn_level High level qualifier used to build data set names for data files.

MANAGEMENTCLASS

The MANAGEMENTCLASS parameter is used to specify the DFSMS management class name that SOLVE:EPS will use for allocation of the data files.

There is no default.

The alias for MANAGEMENTCLASS is MGMTCLAS.

SET SPOOL MANAGEMENTCLASS *mgmt_clas*

Syntax Description

mgmt_clas The DFSMS management class used to allocate data files.

Read the *MVS/ESA SP V5 JCL Reference* and the *MVS/ESA SP V5 JCL User's Guide* for more information.

READBUFNO

Sets the number of buffers used by an Outprocessor for the Basic Sequential Access Method (BSAM) in reading the data files.

SET SPOOL READBUFNO *buffers*

Syntax Description

buffers Number of buffers to be allocated.
Default is 5.

PRIMARYSPACE

The PRIMARYSPACE parameter specifies the primary space allocation SOLVE:EPS will use for the data files.

The default is 10.

SET SPOOL PRIMARYSPACE *space*

Syntax Description

space Primary amount of space to allocate.

Read the *MVS/ESA SP V5 JCL Reference* and the *MVS/ESA SP V5 JCL User's Guide* for more information.

SECONDARYSPACE

The SECONDARYSPACE parameter specifies the secondary space allocation SOLVE:EPS will use for the data files.

The default is 20.

SET SPOOL SECONDARYSPACE *space*

Syntax Description

space Secondary amount of space to allocate.

Read the *MVS/ESA SP V5 JCL Reference* and the *MVS/ESA SP V5 JCL User's Guide* for more information.

STORAGECLASS

The STORAGECLASS parameter specifies the DFSMS storage class name that SOLVE:EPS will use for allocation of the data files.

There is no default.

The alias for STORAGECLASS is STORCLAS.

SET SPOOL STORAGECLASS *stor_clas*

Syntax Description

stor_clas The DFSMS storage class used to allocate data files.

Read the *MVS/ESA SP V5 JCL Reference* and the *MVS/ESA SP V5 JCL User's Guide* for more information.

UNIT

The UNIT parameter specifies the unit name that SOLVE:EPS will use for allocation of the data files.

The default is SYSDA.

SET SPOOL UNIT *unit*

Syntax Description

unit Esoteric unit name used to allocate data files.

Read the *MVS/ESA SP V5 JCL Reference* and the *MVS/ESA SP V5 JCL User's Guide* for more information.

WRITEBUFNO

Sets the number of buffers used by an Inprocessor for the Basic Sequential Access Method (BSAM) in writing the data files.

SET SPOOL WRITEBUFNO *buffers*

Syntax Description

<i>buffers</i>	Number of buffers to be allocated. Default is 5.
----------------	---

SHOW ALTERNATECONSOLE

The **SHOW ALTERNATECONSOLE** command displays the setting for a given alternate console.

Note: No parameters are required for the **SHOW ALTERNATECONSOLE** command. However, if used, the order of the syntax is important.

CONSOLE/NOCONSOLE and LOG/NOLOG, if used, must precede all other parameters.

The ALL parameter includes COUNTERS, SESSIONS, SETTINGS, and STATUS parameters. If used, ALL ends the syntax.

```
SHOW ALTERNATECONSOLE cons_name  
[ CONSOLE | NOCONSOLE ] [ LOG | NOLOG ] [ EVERY ]  
[ ALL ] [ COUNTERS ] [ SESSION ] [ SETTINGS ] [ STATUS ]
```

The default is STATUS.

Aliases for ALTERNATECONSOLE are ACN or ALTCONS.

Syntax Description

CONSOLE NOCONSOLE	CONSOLE displays the output to the console; NOCONSOLE displays no output to the console. CONSOLE is the default.
LOG NOLOG	LOG displays the output to the log; NOLOG displays no output to the log. NOLOG is the default.
EVERY	When the SETTINGS parameter is specified, EVERY shows every possible setting, including those not set.
ALL	Show all information about this console.
COUNTERS	Shows alternate console statistics.
SESSIONS	Shows all sessions that an alternate console currently established.
SETTINGS	Display the values of the settable parameters on this console.
STATUS	Display the status of this console.

This is an example of a **SHOW ALTERNATECONSOLE** command with the EVERY and ALL parameters set:

```
T07MX617S TCP Alternate Console 'ECONTROL' is stopped
T07MX618S ..Current connections:0; limit:16
T07MX699S ..
T07MX600S Settings for TCP Alternate Console 'ECONTROL':
T07MX601S ..Message level           = noinformation, error
T07MX601S ..Log level               = debug, warning
T07MX601S ..Trace word              = 0x00000000
T07MX601S ..Environment member     = ""
T07MX601S ..Upper case              = no
T07MX601S ..Response filter         = ""
T07MX601S ..Listen port             = 1716
T07MX601S ..TCP/IP timeout          = 15
T07MX601S ..TCP/IP retry interval   = 10
T07MX601S ..Maximum sessions        = 16
T07MX601S ..Send buffer size         = 8192
T07MX601S ..Translation table       = "ENGLISH"
T07MX601S ..TCP/IP system ID        = "ACSX"
T07MX601S ..Host table              = "T07HST01"
T07MP000R Command 'SHOW ALTERNATECONSOLE ECONTROL' completed
```

SHOW ALTERNATELOG

The **SHOW ALTERNATELOG** command displays the setting for a given alternate log.

SHOW ALTERNATELOG *log_name*

[**CONSOLE** | **NOCONSOLE**] [**LOG** | **NOLOG**] [**EVERY**]
[**ALL** | [**SETTINGS**] [**STATUS**]]

The default is STATUS.

Aliases for ALTERNATELOG are ALG or ALTLOG.

Syntax Description

CONSOLE NOCONSOLE	CONSOLE displays the output to the console; NOCONSOLE displays no output to the console. CONSOLE is the default.
LOG NOLOG	LOG displays the output to the log; NOLOG displays no output to the log. NOLOG is the default.
EVERY	When the SETTINGS parameter is specified, EVERY shows every possible setting, including those not set.
ALL	Show all information about this log.
SETTINGS	Display the values of the settable parameters on this log.
STATUS	Display the status of this log.

This is an example of a **SHOW ALTERNATELOG** command with the EVERY and ALL parameters set:

```
T07MX617S e-Control Alternate Log 'ECONTROL' is stopped
T07MX699S ..
T07MX600S Settings for e-Control Alternate Log 'ECONTROL':
T07MX601S ..Message level          = information, warning
T07MX601S ..Log level              = information, warning
T07MX601S ..Trace word             = 0x00000000
T07MX601S ..Environment member     = ""
T07MX601S ..Message filter         = ""
T07MX601S ..e-Control system ID    = "OMXC"
T07MX601S ..Retry interval         = 30
T07MP000R Command 'SHOW ECONTROL' completed
```

SHOW CONVERTER

The **SHOW CONVERTER** command displays the settings for a given CONVERTER object.

SHOW CONVERTER *conv_name*

Syntax Description

<i>conv_name</i>	Name of the CONVERTER object whose settings are to be shown. If this option is an asterisk, all defined CONVERTER objects are shown.
------------------	--

SHOW EPS

The **SHOW EPS** command displays information about the whole SOLVE:EPS product.

Aliases for **SHOW EPS** are DISPLAY EPS, LIST EPS, and QUERY EPS.

Note: No parameters are required for the **SHOW EPS** command. However, if used, the order of the syntax is important.

CONSOLE/NOCONSOLE and LOG/NOLOG, if used, must precede all other parameters.

The ALL parameter includes COUNTERS, STATUS, SETTINGS, and OBJECTS parameters. If used, ALL ends the syntax.

The OBJECTS parameter includes INPROCESSORS, OUTPROCESSORS, and QUEUES. If used, OBJECTS ends the syntax.

SHOW EPS

```
[
  [ CONSOLE | NOCONSOLE ] [ LOG | NOLOG ] [ EVERY ] [ MODEL ]
  [ ALL ] [ [ COUNTERS ] [ STATUS ] [ SETTINGS ] ]
  [ OBJECTS ] [ [ INPROCESSORS ] [ OUTPROCESSORS ] [ QUEUES ] ]
]
```

Syntax Description

CONSOLE NOCONSOLE	CONSOLE displays the output to the console; NOCONSOLE displays no output to the console. CONSOLE is the default.
LOG NOLOG	LOG displays the output to the log; NOLOG displays no output to the log. NOLOG is the default.
EVERY	When the SETTINGS parameter is specified, EVERY shows every possible setting, including those not set.
MODEL	Shows all models used for objects on this SHOW EPS command.
ALL	Show all information about SOLVE:EPS.
COUNTERS	Show a set of job counters for SOLVE:EPS jobs.
STATUS	Display the status of SOLVE:EPS. STATUS is the default.
SETTINGS	Display the values of the settable parameters in SOLVE:EPS.
OBJECTS	Show summary information about all objects (Inprocessors, Outprocessors, queues).
INPROCESSORS	Show summary information about all Inprocessors.

OUTPROCESSORS Show summary information about all Outprocessors.

QUEUES Show summary information about all queues.

This is an example of a **SHOW EPS** command with the STATUS parameter set:

```
T07M0610S EPS V5.2 is available; started Fri Jun 19 11:47:15 1998_
T07MS610S ..Primary checkpoint interface is enabled
T07MS610S ..Backup checkpoint interface is enabled
T07MS611S ..Jobs__ initial:0, ready:0, active:0,
T07MS612S ..      held:0, purging:0; total:0
T07MX610S ..INPs   starting:0, available:4,
T07MX611S ..      stopping:0, stopped:0; total:4
T07MX610S ..OUTPs  starting:0, available:2,
T07MX611S ..      stopping:0, stopped:0; total:2
T07MX610S ..ALTCNs starting:0, available:0,
T07MX611S ..      stopping:0, stopped:2; total:2
T07MX610S ..ALTLGs starting:0, available:0,
T07MX611S ..      stopping:0, stopped:0; total:0
T07MQ610S ..Queues retrying:0, available:5,
T07MQ611S ..      stopping:0, stopped:0; total:5
T07MG610S ..Purge task is available
T07ML610S ..Log task is available; logging is active
T07MV602S ..Converter count:7
T07MP000R Command 'SHOW EPS' completed
```

This is an example of a **SHOW EPS** command with EVERY and ALL parameters set:

```
T07M0610S EPS V5.2 is available; started Fri Jun 19 11:47:15 1998
T07MS610S ..Primary checkpoint interface is enabled
T07MS610S ..Backup checkpoint interface is enabled
T07MS611S ..Jobs   initial:0, ready:0, active:0,
T07MS612S ..      held:0, purging:0; total:0
T07MX610S ..INPs   starting:0, available:4,
T07MX611S ..      stopping:0, stopped:0; total:4
T07MX610S ..OUTPs  starting:0, available:2,
T07MX611S ..      stopping:0, stopped:0; total:2
T07MX610S ..ALTCNs starting:0, available:0,
T07MX611S ..      stopping:0, stopped:2; total:2
T07MX610S ..ALTLGs starting:0, available:0,
T07MX611S ..      stopping:0, stopped:0; total:0
T07MQ610S ..Queues retrying:0, available:5,
T07MQ611S ..      stopping:0, stopped:0; total:5
T07MG610S ..Purge task is available
T07ML610S ..Log task is available; logging is active
T07MV602S ..Converter count:7
T07M0699S ..
T07M0600S Settings for EPS:
T07M0601S ..Message level      = information, error
T07M0601S ..Log level         = information, error
T07M0601S ..Trace word        = 0x00000000
T07M0601S ..Upper case       = no
T07M0601S ..Lower case indicator = ""
T07M0699S ..
```

```

T07MX613S Summary of EPS INPs:                Type State
T07MX614S ..-----
T07MX615S ..BSD                               BSD   available
T07MX615S ..NJE                               NJE   available
T07MX615S ..SSI                               SSI   available
T07MX615S ..V                                 VPRT  available
T07M0699S ..
T07MX613S Summary of EPS OUTPs:               Type State
T07MX614S ..-----
T07MX615S ..BSD                               BSD   available
T07MX615S ..NJE                               NJE   available
T07M0699S ..
T07MX613S Summary of EPS ALTCONSS:            Type State
T07MX614S ..-----
T07MX615S ..ECONTROL                           TCP   stopped
T07MX615S ..TELNET                             TCP   stopped
T07M0699S ..
T07MX616S There are no ALTLOGs defined
T07M0699S ..
T07MQ613S Summary of EPS Queues:              Type State
T07MQ614S ..-----
T07MQ615S ..any                               NJE_  available
T07MQ615S ..ARIEL                             BSD_  available
T07MQ615S ..HPJET                             BSD_  available
T07MQ615S ..MFP                               NJE_  available
T07MQ615S ..VOID                             BSD_  available
T07M0699S ..
T07M0611S Counters for EPS:
T07M0612S ..      Job I/O counters -----
T07M0613S ....      0      0 Interface reads_ / bytes
T07M0613S ....      0      0 Interface writes / bytes
T07M0613S ....      0      0 Spool      reads_ / bytes
T07M0613S ....      0      0 Spool      writes / bytes
T07M0614S ..      0_ Job creations attempted -----
T07M0615S ....      0 Created
T07M0615S ....      0 Stopped
T07M0615S ....      0 Incomplete
T07M0616S ..      0_ Job dispatches -----
T07M0617S ....      0 Completed
T07M0617S ....      0 Reset
T07M0617S ....      0 Failed
T07MP000R Command 'SHOW EPS' completed

```

SHOW INPROCESSOR

The **SHOW INPROCESSOR** command displays information about a particular Inprocessor or all Inprocessors.

Aliases for **SHOW INPROCESSOR** are DISPLAY INPROCESSOR, LIST INPROCESSOR, and QUERY INPROCESSOR.

Note: No parameters are required for the **SHOW INPROCESSOR** command. However, if used, the order of the syntax is important.

CONSOLE/NOCONSOLE and LOG/NOLOG, if used, must precede all other parameters.

The ALL parameter includes COUNTERS, SETTINGS, and STATUS parameters. If used, ALL ends the syntax.

```
SHOW INPROCESSOR inp_name | *
[
  [ CONSOLE | NOCONSOLE ] [ LOG | NOLOG ] [ EVERY ] [ MODEL ]
  [ ALL ] | [ [ COUNTERS ] [ SETTINGS ] [ STATUS ] ]
]
```

Syntax Description

<i>inp_name</i>	Name of the Inprocessor object to show. You can use the asterisk (*) to signify all Inprocessors.
CONSOLE NOCONSOLE	CONSOLE displays the output to the console; NOCONSOLE displays no output to the console. CONSOLE is the default.
LOG NOLOG	LOG displays the output to the log; NOLOG displays no output to the log. NOLOG is the default.
EVERY	When the SETTINGS parameter is specified, EVERY shows every possible setting, even if not set.
MODEL	Display models. This is useful only when SHOW INPROCESSOR * is used (in other words, when you want to see all Inprocessors that are models).
ALL	Show all information about the Inprocessor (COUNTERS, SETTINGS, and STATUS).
COUNTERS	Show the job I/O and completion counters for SHOW INPROCESSOR .
SETTINGS	Display all fields of the Inprocessor object as shown in the SET INPROCESSOR command.
STATUS	Display the status of the Inprocessor. Fields to display are <i>inp_name</i> , <i>type</i> , <i>status</i> , <i>sessions</i> (active and total). STATUS is the default.

This is an example of a **SHOW INPROCESSOR** command with the EVERY and ALL parameters set:

```

T07MX617S NJE Inprocessor 'NJE' is available
T07MX618S ..Current sessions:0; limit:1
T07MX699S ..
T07MX600S Settings for NJE Inprocessor 'NJE':
T07MX601S ..Message level           = information, warning
T07MX601S ..Log level               = information, warning
T07MX601S ..Trace word              = 0x00000000
T07MX601S ..Environment member     = ""
T07MX601S ..Maximum sessions       = 1
T07MX601S ..Default queue          = ""
T07MX601S ..Translation             = UNSPECIFIED
T07MX601S ..Priority                = 0
T07MX601S ..NJE VTAM APPLID        = "EPSTS3LU"
T07MX601S ..NJE node name          = "EPSTEST3"
T07MX601S ..NJE VTAM password      = ""
T07MX601S ..NJE partner            = "JES2"
T07MX601S ..JES VTAM APPLID        = "ICSJES"
T07MX601S ..JES NJE node name      = "ICS"
T07MX601S ..JES NJE password       = ""
T07MX601S ..UCS table member       = ""
T07MX601S ..NJE table member       = ""
T07MX601S ..Job name               = ""
T07MX601S ..Job accounting info     = ""
T07MX601S ..Job programmer name     = ""
T07MX601S ..Job notify user ID      = ""
T07MX601S ..Job room number         = ""
T07MX601S ..Job output LINECT       = not set
T07MX601S ..Job output CLASS        = ""
T07MX601S ..Job output DEST         = ""
T07MX601S ..Job output FORMS        = ""
T07MX601S ..Job output WRITER       = ""
T07MX601S ..Job output HOLD         = not set
T07MX601S ..Job output FCB          = ""
T07MX601S ..Job output UCS          = ""
T07MX601S ..Job output PRMODE       = ""
T07MX601S ..Job output TRC          = not set
T07MX601S ..Job output CHARS(1)     = ""
T07MX601S ..Job output CHARS(2)     = ""
T07MX601S ..Job output CHARS(3)     = ""
T07MX601S ..Job output CHARS(4)     = ""
T07MX601S ..Job output FLASH        = ""
T07MX601S ..Job output PAGEDEF      = ""
T07MX601S ..Job output FORMDEF      = ""
T07MX601S ..Job output indentation = not set
T07MX601S ..Job output page width_ = not set
T07MX699S ..
T07MX627S Counters for NJE Inprocessor 'NJE':
T07MX628S ..          0  Created Jobs -----
T07MX629S ....          0          0  Interface reads / bytes
T07MX629S ....          0          0  Interface writes / bytes
T07MX629S ....          0          0  Spool      reads / bytes
T07MX629S ....          0          0  Spool      writes / bytes

```

```
T07MX628S ..      0  Stopped Jobs (operator intervention) ---
T07MX629S ....      0      0  Interface reads / bytes
T07MX629S ....      0      0  Interface writes / bytes
T07MX629S ....      0      0  Spool      reads / bytes
T07MX629S ....      0      0  Spool      writes / bytes
T07MX628S ..      0  Incomplete Jobs -----
T07MX629S ....      0      0  Interface reads / bytes
T07MX629S ....      0      0  Interface writes / bytes
T07MX629S ....      0      0  Spool      reads / bytes
T07MX629S ....      0      0  Spool      writes / bytes
T07MX628S ..      0  Attempted Job creations total -----
T07MX629S ....      0      0  Interface reads / bytes
T07MX629S ....      0      0  Interface writes / bytes
T07MX629S ....      0      0  Spool      reads / bytes
T07MX629S ....      0      0  Spool      writes / bytes
T07MP000R Command 'SHOW INPROCESSOR NJE' completed
```

SHOW JOB

The **SHOW JOB** command displays information about a particular SOLVE:EPS job or selected SOLVE:EPS jobs.

Aliases for **SHOW JOB** are DISPLAY JOB, LIST JOB, and QUERY JOB.

Note: No parameters are required for the **SHOW JOB** command. However, if used, the order of the syntax is important.

CONSOLE/NOCONSOLE and LOG/NOLOG, if used, must precede all other parameters.

The ALL parameter includes ATTRIBUTES, COUNTERS, SETTINGS, and STATUS parameters. If used, ALL ends the syntax.

```
SHOW JOB eps_job_id1 [ eps_job_id2 ... | THRU eps_job_idn ]
[
  [ CONSOLE | NOCONSOLE ] [ LOG | NOLOG ] [ EVERY ]
  [ ALL ] | [ [ ATTRIBUTES ] [ COUNTERS ] [ SETTINGS ] [ STATUS ] ]
]
```

Syntax Description

<i>eps_job_id</i>	The SOLVE:EPS job identification. You can use the asterisk (*) to signify all jobs. <i>eps_job_id</i> is of the form the character J followed by up to five numerics. Therefore, <i>eps_job_id</i> can range from J00001 to J99999. Leading zeros for the numerics can be omitted, so J00001 is the same as J1.
THRU	Range of jobs to be held. Range is 1 to 99999.
CONSOLE NOCONSOLE	CONSOLE displays the output to the console; NOCONSOLE displays no output to the console. CONSOLE is the default.
LOG NOLOG	LOG displays the output to the log; NOLOG displays no output to the log. NOLOG is the default.
EVERY	When the SETTINGS parameter is specified, EVERY shows every possible setting, including those not set.
ALL	Show all information about the job (ATTRIBUTES, COUNTERS, SETTINGS, and STATUS).
ATTRIBUTES	Display all nonsettable attributes for the job(s).
COUNTERS	Show a set of job counters.
SETTINGS	Display all settable attributes for the job(s).

STATUS Display the status of the job(s). Fields displayed can be *queue*, *status* (inact, active, operhold, retryhold, sent), *origin*, and some JES defaults. STATUS is the default.

This is an example of the **SHOW JOB** command with the EVERY and ALL parameters set:

```

T07MJ610S Job J00246(MFP1) is ready
T07MJ613S ..Queue 'VOID' is stopped
T07MJ614S ..BSD OUTP 'BSD' is available
T07MJ618S ..Priority:1; create time:1998.170 12:02:36.79
T07MJ619S ..Times completed:0, reset:0, failed:0
T07MJ699S ..
T07MJ600S Settings for Job J00246(MFP1):
T07MJ601S ..Queue name           = "VOID"
T07MJ601S ..Priority              = 1
T07MJ601S ..Job disposition       = purge
T07MJ601S ..Job output LINECT     = not set
T07MJ601S ..Job output CLASS      = "A"
T07MJ601S ..Job output DEST       = "EPSTEST3.VOID"
T07MJ601S ..Job output FORMS      = ""
T07MJ601S ..Job output WRITER     = ""
T07MJ601S ..Job output HOLD       = not set
T07MJ601S ..Job output FCB        = ""
T07MJ601S ..Job output UCS        = ""
T07MJ601S ..Job output PRMODE     = "LINE"
T07MJ601S ..Job output TRC        = not set
T07MJ601S ..Job output CHARS(1)   = ""
T07MJ601S ..Job output CHARS(2)   = ""
T07MJ601S ..Job output CHARS(3)   = ""
T07MJ601S ..Job output CHARS(4)   = ""
T07MJ601S ..Job output FLASH      = ""
T07MJ601S ..Job output PAGEDEF    = ""
T07MJ601S ..Job output FORMDEF    = ""
T07MJ601S ..Job output indentation = not set
T07MJ601S ..Job output page width = not set
T07MJ699S ..
T07MJ608S Attributes for Job J00246(MFP1):
T07MJ609S ..Status                 = ready
T07MJ609S ..INProcessor name       = "NJE"
T07MJ609S ..INProcessor type       = NJE
T07MJ609S ..Create time            = 1998.170 12:02:36.79
T07MJ609S ..Number of datasets     = 1
T07MJ609S ..Source data descriptor = IBM machine
T07MJ609S ..Current dataset        = 0
T07MJ609S ..Job name               = "MFP1"
T07MJ609S ..Job notify user ID     = "MFP1"
T07MJ609S ..JES job number         = 1331
T07MJ609S ..JES job class          = \
T07MJ609S ..JES job message class_ = Z
T07MJ609S ..JES job estimated line = 999999000
T07MJ609S ..Job origin node        = "ICS"
T07MJ609S ..Job origin user        = "MFP1"
T07MJ609S ..Job user area (+00)     = 0x00000000000000000000000000000000
T07MJ609S ..Job user area (+10)     = 0x00000000000000000000000000000000

```

```

T07MJ609S ...Job user area (+20)    = 0x00000000000000000000000000000000
T07MJ609S ...Job user area (+30)    = 0x00000000000000000000000000000000
T07MJ699S ..
T07MJ620S Counters for Job J00246(MFP1):
T07MJ621S ..          Current I/O -----
T07MJ622S ....          0          0 Interface reads / bytes
T07MJ622S ....          0          0 Interface writes / bytes
T07MJ622S ....          0          0 Spool      reads / bytes
T07MJ622S ....          0          0 Spool      writes / bytes
T07MJ621S ..          0 Times completed -----
T07MJ622S ....          0          0 Interface reads / bytes
T07MJ622S ....          0          0 Interface writes / bytes
T07MJ622S ....          0          0 Spool      reads / bytes
T07MJ622S ....          0          0 Spool      writes / bytes
T07MJ621S ..          0 Times reset -----
T07MJ622S ....          0          0 Interface reads / bytes
T07MJ622S ....          0          0 Interface writes / bytes
T07MJ622S ....          0          0 Spool      reads / bytes
T07MJ622S ....          0          0 Spool      writes / bytes
T07MJ621S ..          0 Times failed -----
T07MJ622S ....          0          0 Interface reads / bytes
T07MJ622S ....          0          0 Interface writes / bytes
T07MJ622S ....          0          0 Spool      reads / bytes
T07MJ622S ....          0          0 Spool      writes / bytes
T07MJ621S ..          0 Total times dispatched -----
T07MJ622S ....          0          0 Interface reads / bytes
T07MJ622S ....          0          0 Interface writes / bytes
T07MJ622S ....          0          0 Spool      reads / bytes
T07MJ622S ....          0          0 Spool      writes / bytes
T07MJ621S ..          Creation I/O -----
T07MJ622S ....          0          0 Interface reads / bytes
T07MJ622S ....          0          0 Interface writes / bytes
T07MJ622S ....          0          0 Spool      reads / bytes
T07MJ622S ....          49          1200 Spool      writes / bytes
T07MP000R Command 'SHOW JOB' completed

```

SHOW LOG

The **SHOW LOG** command displays the current LOG settings.

Aliases for **SHOW LOG** are DISPLAY LOG, LIST LOG, and QUERY LOG.

Note: No parameters are required for the **SHOW LOG** command. However, if used, the order of the syntax is important.

CONSOLE/NOCONSOLE and LOG/NOLOG, if used, must precede all other parameters.

The ALL parameter includes COUNTERS, SETTINGS, and STATUS parameters. If used, ALL ends the syntax.

SHOW LOG

```
[
  [ CONSOLE | NOCONSOLE ] [ LOG | NOLOG ] [ EVERY ]
  [ ALL ] | [ [ COUNTERS ] [ SETTINGS ] [ STATUS ] ]
]
```

Syntax Description

CONSOLE NOCONSOLE	Display the output to the console; NOCONSOLE displays no output to the console. CONSOLE is the default.
LOG NOLOG	LOG displays the output to the log; NOLOG displays no output to the log. NOLOG is the default.
EVERY	When the SETTINGS parameter is specified, EVERY shows every possible setting, even if not set.
ALL	Show all information about the log. The messages displayed by this command show all possible values for the LOG object (COUNTERS, SETTINGS, and STATUS). All values shown are changeable via the SET LOG command.
COUNTERS	Show a set of log counters.
SETTINGS	Display all the fields of the log task as shown in the SET LOG command.
STATUS	Display the status of the log task. STATUS is the default.

This is an example of the **SHOW LOG** command with the EVERY and ALL parameters set:

```
T07MP002I SHOW LOG EVERY ALL
T07ML611S Status for Log:
T07ML610S ..Log task is available; logging is active
T07ML612S ..Current entries: 1079; total entries: 1762
T07ML613S ..Spin count: 4; next spin time: none scheduled
T07ML699S ..
T07ML600S Settings for Log:
```

```
T07ML601S ..Message level      = information, warning
T07ML601S ..Log level          = information, warning
T07ML601S ..Trace word         = 0x00000000
T07ML601S ..JES output class   = "A"
T07ML601S ..Output forms       = ""
T07ML601S ..JES destination    = ""
T07ML601S ..Spin records       = 0
T07ML601S ..Spin time          = 0 minutes
T07ML601S ..Synchronize spin   = no
T07ML601S ..Upper case         = no
T07MP000R Command 'SHOW LOG' completed
```

SHOW OUTPROCESSOR

The **SHOW OUTPROCESSOR** command displays information about a particular Outprocessor or about all Outprocessors.

Aliases for **SHOW OUTPROCESSOR** are **DISPLAY OUTPROCESSOR**, **LIST OUTPROCESSOR**, and **QUERY OUTPROCESSOR**.

Note: No parameters are required for the **SHOW OUTPROCESSOR** command. However, if used, the order of the syntax is important.

CONSOLE/NOCONSOLE and LOG/NOLOG, if used, must precede all other parameters.

The ALL parameter includes COUNTERS, JOBS, QUEUES, SETTINGS, and STATUS parameters. If used, ALL ends the syntax.

```
SHOW OUTPROCESSOR outp_name | *  
[  
  [ CONSOLE | NOCONSOLE ] [ LOG | NOLOG ] [ EVERY ] [ MODEL ]  
  [ ALL ] | [ [ COUNTERS ] [ JOBS ] [ QUEUES ] [ SETTINGS ] [ STATUS ] ]  
]
```

Syntax Description

<i>outp_name</i>	Name of the Outprocessor object to show. You can use the asterisk (*) to signify all Outprocessors.
CONSOLE NOCONSOLE	CONSOLE displays the output to the console; NOCONSOLE displays no output to the console. CONSOLE is the default.
LOG NOLOG	LOG Display the output to the log; NOLOG displays no output to the log. NOLOG is the default.
MODEL	Display models. This is useful only when SHOW OUTPROCESSOR * is used (in other words, when you want to see all Outprocessors that are models).
EVERY	When the SETTINGS parameter is specified, EVERY shows every possible setting, even if not set.
ALL	Show all information about the Outprocessor (COUNTERS, JOBS, QUEUES, SETTINGS, and STATUS).
COUNTERS	Show the job I/O and completion counters for the Outprocessor object.
JOBS	Show information about jobs active on the Outprocessor.
QUEUES	Show all queues that referenced this Outprocessor.

STATUS	Display the status of the Outprocessor. Fields to display are outp_name, type, status, sessions (active and total). STATUS is the default.
SETTINGS	Display all the fields of the Outprocessor object as shown in the SET OUTPROCESSOR command.

This is an example of the **SHOW OUTPROCESSOR** command with the EVERY and ALL parameters set:

```

T07MX617S BSD Outprocessor 'BSD' is available
T07MX618S ..Active Jobs:0; limit:10
T07MX699S ..
T07MX600S Settings for BSD Outprocessor 'BSD':
T07MX601S ..Message level          = information, warning
T07MX601S ..Log level              = information, warning
T07MX601S ..Trace word             = 0x00000000
T07MX601S ..Environment member     = ""
T07MX601S ..Maximum sessions      = 10
T07MX601S ..Converter Select exit  = ""
T07MX601S ..TCP/IP system ID       = "ACSX"
T07MX699S ..
T07MX623S Summary of Queues for BSD Outprocessor 'BSD':
T07MX624S ..Queue                      State                      #Act
T07MX625S ..-----
T07MX626S ..ARIEL                      available                  .
T07MX626S ..HPJET                      available                  .
T07MX626S ..VOID                      available                  .
T07MX699S ..
T07MX619S There are no Jobs on BSD Outprocessor 'BSD'
T07MX699S ..
T07MX627S Counters for BSD Outprocessor 'BSD':
T07MX628S ..          0 Completed Jobs -----
T07MX629S ....          0          0 Interface reads / bytes
T07MX629S ....          0          0 Interface writes / bytes
T07MX629S ....          0          0 Spool      reads / bytes
T07MX629S ....          0          0 Spool      writes / bytes
T07MX628S ..          0 Reset Jobs -----
T07MX629S ....          0          0 Interface reads / bytes
T07MX629S ....          0          0 Interface writes / bytes
T07MX629S ....          0          0 Spool      reads / bytes
T07MX629S ....          0          0 Spool      writes / bytes
T07MX628S ..          0 Failed Jobs -----
T07MX629S ....          0          0 Interface reads / bytes
T07MX629S ....          0          0 Interface writes / bytes
T07MX629S ....          0          0 Spool      reads / bytes
T07MX629S ....          0          0 Spool      writes / bytes
T07MX628S ..          0 Job dispatches total -----
T07MX629S ....          0          0 Interface reads / bytes
T07MX629S ....          0          0 Interface writes / bytes
T07MX629S ....          0          0 Spool      reads / bytes
T07MX629S ....          0          0 Spool      writes / bytes
T07MP000R Command 'SHOW OUTPROCESSOR BSD' completed

```

SHOW QUEUE

The **SHOW QUEUE** command displays information about a particular queue entry or all entries.

Aliases for **SHOW QUEUE** are **DISPLAY QUEUE**, **LIST QUEUE**, and **QUERY QUEUE**.

Note: No parameters are required for the **SHOW QUEUE** command. However, if used, the order of the syntax is important.

CONSOLE/NOCONSOLE and LOG/NOLOG, if used, must precede all other parameters.

The ALL parameter includes COUNTERS, JOBS, SETTINGS, and STATUS parameters. If used, ALL ends the syntax.

```
SHOW QUEUE q_name | *  
[  
  [ CONSOLE | NOCONSOLE ] [ LOG | NOLOG ] [ EVERY ] [ MODEL ]  
  [ ALL ] | [ [ COUNTERS ] [ JOBS ] [ SETTINGS ] [ STATUS ] ]  
]
```

Syntax Description

<i>q_name</i>	Name of the queue to show. You can use the asterisk (*) to signify all queues.
CONSOLE NOCONSOLE	CONSOLE displays the output to the console; NOCONSOLE displays no output to the console. CONSOLE is the default.
LOG NOLOG	LOG displays the output to the log; NOLOG displays no output to the log. NOLOG is the default.
EVERY	When the SETTINGS parameter is specified, EVERY shows every possible setting, even if not set.
MODEL	Display models. This is useful only with SHOW QUEUE * (in other words, when you want to see all queues that are models).
ALL	Show all information about the queue (COUNTERS, JOBS, SETTINGS, and STATUS).
COUNTERS	Show a set of job counters for this queue.
JOBS	Show information about jobs on the queue.
SETTINGS	Display all the fields of the queue object as shown in the SET QUEUE command.
STATUS	Display the status of the queue. STATUS is the default.

Following is an example of the **SHOW QUEUE** command with the EVERY and ALL parameters set:

```

T07MQ618S Queue 'ARIEL' is available
T07MQ622S ..Jobs active:0, ready:0, held:0; total:0
T07MQ632S ..Active job limit:no limit
T07MQ623S ..BSD OUTP 'BSD' is available
T07MQ633S ..Remote machine IP address:unresolved
T07MQ699S ..
T07MQ600S Settings for Queue 'ARIEL':
T07MQ601S ..Message level          = information, warning
T07MQ601S ..Log level                  = information, warning
T07MQ601S ..Trace word                   = 0x00000000
T07MQ601S ..Description                 = ""
T07MQ601S ..OUTProcessor name         = "BSD"
T07MQ601S ..Conversion cache size      = 0
T07MQ601S ..Maximum sessions            = -1
T07MQ601S ..Retry interval               = 300 seconds
T07MQ601S ..Retry threshold              = 48
T07MQ601S ..Job disposition              = purge
T07MQ601S ..Supported translation        = UNSPECIFIED
T07MQ601S ..Maximum line size            = 255
T07MQ601S ..Maximum page size            = 255
T07MQ601S ..No last form feed            = no
T07MQ601S ..Dispatch user exit           = ""
T07MQ601S ..Dataset user exit            = ""
T07MQ601S ..Local machine name           = resolved by outprocessor
T07MQ601S ..Local port(s)                = 721 thru 731
T07MQ601S ..Local port delay timer       = 5 seconds
T07MQ601S ..Remote machine name          = "PISCES"
T07MQ601S ..Remote printer name          = "ariel"
T07MQ601S ..Remote port                  = 515
T07MQ601S ..Control file line exit       = ""
T07MQ601S ..Control file position        = last
T07MQ601S ..No banner page               = no
T07MQ601S ..Indentation                  = not set
T07MQ601S ..Page width                   = not set
T07MQ601S ..Filter                       = "f"
T07MQ601S ..Control string prefix        = not set
T07MQ601S ..Control string suffix        = not set
T07MQ601S ..FileName TimeStamp          = yes
T07MQ601S ..Send All Copies             = no
T07MQ699S ..
T07MQ624S There are no Jobs on Queue 'ARIEL'
T07MQ699S ..
T07MQ629S Counters for Queue 'ARIEL':
T07MQ630S ..          0 Completed Jobs -----
T07MQ631S ....          0          0 Interface reads / bytes
T07MQ631S ....          0          0 Interface writes / bytes
T07MQ631S ....          0          0 Spool      reads / bytes
T07MQ631S ....          0          0 Spool      writes / bytes
T07MQ630S ..          0 Reset Jobs -----
T07MQ631S ....          0          0 Interface reads / bytes
T07MQ631S ....          0          0 Interface writes / bytes
T07MQ631S ....          0          0 Spool      reads / bytes
T07MQ631S ....          0          0 Spool      writes / bytes
T07MQ630S ..          0 Failed Jobs -----

```

T07MQ631S		0	0	Interface reads / bytes
T07MQ631S		0	0	Interface writes / bytes
T07MQ631S		0	0	Spool reads / bytes
T07MQ631S		0	0	Spool writes / bytes
T07MQ630S	..	0		Job dispatches total -----
T07MQ631S		0	0	Interface reads / bytes
T07MQ631S		0	0	Interface writes / bytes
T07MQ631S		0	0	Spool reads / bytes
T07MQ631S		0	0	Spool writes / bytes
T07MP000R	Command 'SHOW QUEUE ARIEL' completed			

SHOW SPOOL

The **SHOW SPOOL** command displays the current settings for the SOLVE:EPS SPOOL parameters. All values shown are changeable via the **SET SPOOL** command.

Aliases for **SHOW SPOOL** are DISPLAY SPOOL, LIST SPOOL, and QUERY SPOOL.

Note: No parameters are required for the **SHOW SPOOL** command. However, if used, the order of the syntax is important.

CONSOLE/NOCONSOLE and LOG/NOLOG, if used, must precede all other parameters.

The ALL parameter includes ALLJOBS, COUNTERS, STATUS, and SETTINGS parameters. If used, ALL ends the syntax.

ALLJOBS includes JOBS and ORPHANS.

SHOW SPOOL

```
[
  [ CONSOLE | NOCONSOLE ] [ LOG | NOLOG ] [ EVERY ]
  [ ALL ] |
  [
    [ ALLJOBS [ JOBS | ORPHANS ] ] [ COUNTERS ] [ STATUS ] [ SETTINGS ]
  ]
]
```

Syntax Description

CONSOLE NOCONSOLE	CONSOLE displays the output to the console; NOCONSOLE displays no output to the console. CONSOLE is the default
LOG NOLOG	LOG displays the output to the log' MP:PG displays no output to the log. NOLOG is the default.
EVERY	When the SETTINGS parameter is specified, EVERY shows every possible setting, even if not set.
ALL	Show all information about the spool.
ALLJOBS	Show a summary of all jobs in SOLVE:EPS.
COUNTERS	Show job I/O and completion counters for this spool.
STATUS	Display the status of spool.
SETTINGS	Display the values of the settable spool parameters.
JOBS	Show a summary of non-orphaned jobs.
ORPHANS	Show a summary of orphaned jobs.

This is an example of the **SHOW SPOOL** command with the EVERY and ALL parameters set:

```

T07MS613S Status for Spool:
T07MS610S ..Primary checkpoint interface is enabled
T07MS610S ..Backup checkpoint interface is enabled
T07MS611S ..Jobs    initial:0, ready:0, active:0,
T07MS612S ..        held:0, purging:0; total:0
T07MG610S ..Purge task is available
T07MG611S ..Purge status:idle
T07MS699S ..
T07MS600S Settings for Spool:
T07MS601S ..Message level           = information, error
T07MS601S ..Log level               = information, warning
T07MS601S ..Trace word              = 0x00000000
T07MS601S ..High level qualifier    = "MFP1.EPS410.SPOOL"
T07MS601S ..Primary space          = 300
T07MS601S ..Secondary space         = 20
T07MS601S ..Allocation unit         = track
T07MS601S ..Unit                   = "3390"
T07MS601S ..Storage class           = ""
T07MS601S ..Management class        = ""
T07MS601S ..Data class              = ""
T07MS601S ..Average records         = not set
T07MS601S ..Read BUFNO              = 5
T07MS601S ..Write BUFNO             = 5
T07MS601S ..Block size              = 27998
T07MS699S ..
T07MS618S There are no Jobs in the spool
T07MS699S ..
T07MS619S Counters for Spool:
T07MS620S ..Attempted Job creations:
T07MS621S ....      0 Created Jobs -----
T07MS622S .....      0      0 Interface reads / bytes
T07MS622S .....      0      0 Interface writes / bytes
T07MS622S .....      0      0 Spool    reads / bytes
T07MS622S .....      0      0 Spool    writes / bytes
T07MS621S ....      0 Stopped Jobs (operator intervention) ---
T07MS622S .....      0      0 Interface reads / bytes
T07MS622S .....      0      0 Interface writes / bytes
T07MS622S .....      0      0 Spool    reads / bytes
T07MS622S .....      0      0 Spool    writes / bytes
T07MS621S ....      0 Incomplete Jobs -----
T07MS622S .....      0      0 Interface reads / bytes
T07MS622S .....      0      0 Interface writes / bytes
T07MS622S .....      0      0 Spool    reads / bytes
T07MS622S .....      0      0 Spool    writes / bytes
T07MS621S ....      0 Attempted Job creations total -----
T07MS622S .....      0      0 Interface reads / bytes
T07MS622S .....      0      0 Interface writes / bytes
T07MS622S .....      0      0 Spool    reads / bytes
T07MS622S .....      0      0 Spool    writes / bytes
T07MS620S ..Job dispatches:
T07MS621S ....      0 Completed Jobs -----
T07MS622S .....      0      0 Interface reads / bytes

```

T07MS622S		0	0	Interface writes / bytes
T07MS622S		0	0	Spool reads / bytes
T07MS622S		0	0	Spool writes / bytes
T07MS621S		0	Reset Jobs -----	
T07MS622S		0	0	Interface reads / bytes
T07MS622S		0	0	Interface writes / bytes
T07MS622S		0	0	Spool reads / bytes
T07MS622S		0	0	Spool writes / bytes
T07MS621S		0	Failed Jobs -----	
T07MS622S		0	0	Interface reads / bytes
T07MS622S		0	0	Interface writes / bytes
T07MS622S		0	0	Spool reads / bytes
T07MS622S		0	0	Spool writes / bytes
T07MS621S		0	Job dispatches total -----	
T07MS622S		0	0	Interface reads / bytes
T07MS622S		0	0	Interface writes / bytes
T07MS622S		0	0	Spool reads / bytes
T07MS622S		0	0	Spool writes / bytes
T07MP000R Command 'SHOW SPOOL' completed				

SHUTDOWN

The **SHUTDOWN** command terminates the SOLVE:EPS address space.

An MVS **STOP (P)** command can also be used to terminate SOLVE:EPS. It is equivalent to a **SHUTDOWN IMMEDIATE** command.

The default is DRAIN.

SHUTDOWN DRAIN | IMMEDIATE | FORCE

Syntax Description

DRAIN	Currently active sessions and jobs are allowed to complete. The SOLVE:EPS job step will end with a return code of zero. DRAIN is the default.
IMMEDIATE	Currently active sessions and jobs are interrupted. The SOLVE:EPS job step will end with a return code of 4.
FORCE	Shutdown is performed forcibly. SOLVE:EPS does not clean up or close files; it terminates the main program. The SOLVE:EPS job step will end with a return code of 8.

Note: If you must force a shutdown of SOLVE:EPS, using the FORCE parameter will immediately close the checkpoint and log files, which may leave job spool files allocated on DASD. These will be removed by SOLVE:EPS upon restart. If you use the MVS **CANCEL** command instead of **SHUTDOWN FORCE**, the checkpoint and log files might not be closed properly and there could be problems when you restart SOLVE:EPS.

SPIN

The **SPIN** command closes the SOLVE:EPS log and starts a new system output (SYSOUT) log stream. This is useful when the log is becoming large and it is desirable to restart the log into a new SYSOUT. When the spin occurs the currently specified log output parameters are used to allocate the new log.

Note: The term “spinning” describes the action of closing and reopening the log.

SPIN [LOG]

START

The **START** command makes active the various SOLVE:EPS objects. A SOLVE:EPS object must be started before use.

When you start either an NJE Inprocessor or Outprocessor, the other is started automatically.

Note: Queues have several states that make them startable. A queue can be in Stopped, Quiesce (Retry), Autostopping (to be stopped due to exceeded retry count), or Draining state.

START *obj_type obj_name* [**NJE** | **LOGICALUNITNAME** *lu_name*]

Syntax Description

<i>obj_type</i>	The type of object to start. The valid choices are ALTERNATECONSOLE, ALTERNATELOG, INPROCESSOR, OUTPROCESSOR, and QUEUE.
<i>obj_name</i>	Name of the object to be started. You can use an asterisk (*) to start objects of the specified type. If you use the asterisk (*), MODEL objects will be ignored. If the object has been defined with MODEL, it cannot and will not be started.
NJE	This option must be specified when starting an NJE Inprocessor or Outprocessor. It ensures that the related NJE Inprocessor and Outprocessor start together. If not specified, an error results and the object does not start.
LOGICALUNITNAME <i>lu_name</i>	Use this option for an Inprocessor of type VPRT. It specifies the VTAM LU name that the VPRT Inprocessor should start. This option is typically used if VTAM has problems with the logical unit and the ACB (APPLID) needs to be reopened by the VPRT Inprocessor. The alias for LOGICALUNITNAME is LUNAME.

STOP

The **STOP** command makes various SOLVE:EPS objects inactive. When a SOLVE:EPS object is inactive it does not perform any work.

The default is DRAIN.

```
STOP obj_type obj_name DRAIN | IMMEDIATE | FORCE | SESSION sess_num |  
  [ NJE | LOGICALUNITNAME lu_name ]
```

Syntax Description

<i>obj_type</i>	Type of the object to stop. The valid choices are ALTERNATECONSOLE, ALTERNATELOG, INPROCESSOR, OUTPROCESSOR, or QUEUE.
<i>obj_name</i>	Name of the object to be stopped. You can use the asterisk (*) to stop all objects of the specified type.
DRAIN	The object is quiesced and allowed to finish work before becoming inactive. DRAIN is the default.
IMMEDIATE	The object is stopped immediately. All functions of the object terminate.
FORCE	The object is stopped forcibly. All functions of the object are terminated abnormally. This option is not valid for the QUEUE object. NOTE: Using the FORCE option of the STOP command will leave residual memory allocated by the object. Use this parameter with caution.
SESSION <i>sess_num</i>	Stops a particular alternate console or Inprocessor session. The number represents the session number known by the alternate console or Inprocessor.
NJE	This option must be specified when stopping an NJE Inprocessor or Outprocessor. It ensures that the related NJE Inprocessor and Outprocessor stop together. If not specified, an error results and the object does not stop.
LOGICALUNITNAME <i>lu_name</i>	Use this option for an Inprocessor of type VPRT. It specifies the VTAM LU name that the VPRT Inprocessor should start. This option is typically used if VTAM has problems with the logical unit and the ACB (APPLID) needs to be reopened by the VPRT Inprocessor. The alias for LOGICALUNITNAME is LUNAME.

2

SOLVE:EPS Operation

This chapter describes how to operate the SOLVE:EPS (Enterprise Print Services) system. It includes these sections:

- [“SOLVE:EPS Execution” on page 2-2](#)
Describes the JCL used for SOLVE:EPS startup, as well as starting and stopping SOLVE:EPS.
- [“Starting and Stopping SOLVE:EPS” on page 2-8](#)
Describes how to start and stop SOLVE:EPS.
- [“SOLVE:EPS Commands Usage” on page 2-9](#)
Describes the status display and trace control operator commands.
- [“Message and Log Controls” on page 2-12](#)
Describes messages produced by SOLVE:EPS that will be written to the MVS operator console and logged to an SOLVE:EPS designated log.

SOLVE:EPS Execution

This section describes the setup procedures for SOLVE:EPS, and how to start and stop it.

SOLVE:EPS Job Control Language (JCL) Statements

This is an example of the JCL found in T07SAMP member T07EPS:

```
//T07EPS  PROC  INDX='TCPICS.Vxxx',          HI-LEVEL QUAL FOR DSN
//          TCPINDX='IOS390.R1'              TCPACCESS PARM LIBRARY
//          CNFG=T07CFG00                    ,      CONFIG MEMBER
//          SOUT='*'                        SYSOUT CLASS
//*
//*          SAMPLE JCL PROCEDURE TO RUN EPS
//*
//*          EDIT THE INDX, TCPINDX, MEM, AND SOUT SYMBOLIC
//*          PARAMETERS (OR OVERRIDE THEM WHEN EXECUTING
//*          THIS JCL PROCEDURE).
//*
//*          NOTE: SINCE EPS IS INSTALLED IN A SEPARATE TARGET
//*          LOAD LIBRARY FROM TCP, THE TCP LOAD LIBRARY
//*          MUST BE INCLUDED AS A STEPLIB DATASET IN ORDER
//*          MAKE THE SAS/C RUN-TIME LIBRARY AVAILABLE.
//*
//T07EPS    EXEC PGM=T07EPS,REGION=8M,TIME=1440,
//          PARM='&CNFG'
//STEPLIB   DD      DISP=SHR,DSN=&INDX..T07LOAD
//          DD      DISP=SHR,DSN=&TCPINDX..LOAD
//          DD      DISP=SHR,DSN=&TCPINDX..LINK
//SYSPRINT  DD      SYSOUT=&SOUT
//SYSTEM    DD      SYSOUT=&SOUT
//OUTPUT     DD      SYSOUT=&SOUT
//PARMLIB   DD      DISP=SHR,DSN=&INDX..T07PARM
//          DD      DISP=SHR,DSN=&TCPINDX..PARM
//SYSUDUMP   DD      SYSOUT=&SOUT
//EPSCKPTP   DD      DISP=SHR,DSN=&INDX..CKPT
//EPSCKPTS   DD      DISP=SHR,DSN=&INDX..CKPT.BKUP
```

Customizing the JCL

To set up this JCL, this customization is recommended:

- Copy member T07EPS from the T07SAMP data set to a PROCLIB known to JES2. You can change the member name and the started task name to conform to your installation standards.
- Modify the INDX symbolic parameter to reflect the high-level index used in the installation procedure.
- Modify the SOUT symbolic parameter to specify a valid SYSOUT class for your installation.

Note: The default of asterisk (*) may cause the SYSOUT to be flushed when this JCL is executed as a started task.

- Modify the CNFG symbolic parameter to reference the SOLVE:EPS startup EXEC created in the installation procedure.

Changes to the SOLVE:EPS Checkpoint File

Occasionally SOLVE:EPS maintenance is introduced that changes the format of the checkpoint file. This means that after applying the maintenance in question, you must also convert the checkpoint file to the new format in order to run SOLVE:EPS.

There are two ways to handle this. If there are no jobs in the SOLVE:EPS checkpoint file (**SHOW SPOOL ALLJOBS**) after applying the maintenance, you can rerun the checkpoint initialization job to bring the file up to the current level. In addition to changing the checkpoint file level, this procedure also resets the SOLVE:EPS job number, so the next job will be J00001.

If there are jobs in the SOLVE:EPS checkpoint file that you do not want to lose, run the checkpoint file conversion utility. This program updates the checkpoint file level, keeps the SOLVE:EPS job number, and reformats all job and data set records.

Checkpoint File Conversion Utility

The checkpoint file conversion utility, T07SCVCH, converts an existing SOLVE:EPS checkpoint file from one (lower) level to another. You can specify the new level or you can let it default to the current level; the old level is obtained from the existing file.

Conversion is accomplished by reading the “seed” record of the existing checkpoint file, determining the proper conversion from the old and new levels, and then inserting an updated seed record into the new checkpoint file. Then each record is read from the existing checkpoint file, converted, and inserted into the new checkpoint file. Besides the seed record, job records and data set records must also be converted.

Changes From Checkpoint File Level 1 to Checkpoint File Level 2

Specification of Data Translation in Job Records

The primary area of change for a level 2 checkpoint file from a level 1 file is in the specification of data translation in job records. In level 1 job records, translation can be given as a name of a SOLVE:EPS-provided translation table. Level 2 translation is defined by a pair of fields, one describing the source data language and one describing the supported destination data language. When the job is printed, these two values are used to decide how translation is to be done. Since there is no way of correlating a translation table name to source and destination data languages, all jobs will have these two fields set to the default value UNSPECIFIED. (For more information, read the description of the TRANSLATE parameter for one of the operator commands [“REFRESH” on page 1-14](#), [“SET INPROCESSOR” on page 1-31](#), or [“SET QUEUE” on page 1-62](#), or of the TRANSLATE parameters for any of the SOLVE:EPS statements in “SOLVE:EPS Members” (Chapter 8), or the section “Translate Table” in “SOLVE:EPS Members” of the *SOLVE:EPS Installation and Administration Guide*.)

UCS Field

There is also a change in the handling of the UCS field for job records for level 2. If UCS has one of the special values recognized by SOLVE:EPS, it will be interpreted by the T07CSVCH and then set to null. With a level 1 checkpoint file, SOLVE:EPS interpreted the UCS field in the Outprocessor before printing. With a level 2 checkpoint file, SOLVE:EPS interprets and “consumes” the UCS field in the Inprocessor when a SOLVE:EPS job is created. Furthermore, interpretation is no longer hard coded, but specified by a table that NJE and SSI Inprocessors use. (For more information, read “Translate Table” in “SOLVE:EPS Members” (Chapter 8) of the *SOLVE:EPS Installation and Administration Guide*.)

Running the Checkpoint File Conversion Utility

To bring an existing SOLVE:EPS checkpoint file up to a new level, modify the T07SCVUP member of the T07SAMP data set. To set up this JCL, do the following:

- Modify the TCPINDEX value to reflect the high-level index of the SOLVE:TCPaccess load library.
- Modify the INDEX value to reflect the high-level index of SOLVE:EPS data sets.
- Modify the VOLSER1 value to indicate the DASD volume serial for the new primary checkpoint VSAM cluster.
- Modify the VOLSER2 value to indicate the DASD volume serial for the new secondary checkpoint VSAM cluster.

Note: In order to provide adequate recovery in case of a DASD failure, the primary checkpoint cluster and the backup checkpoint cluster should reside on different physical volumes.

You should make a backup of the checkpoint files before running the checkpoint file conversion utility.

Sample T07SCVCH JCL

```

//JOB CARD
//*
//*  SAMPLE JCL TO DEFINE NEW CHECKPOINT FILES AND
//*  AND TO UPDATE CURRENT CHECKPOINT FILES INTO THEM
//*
//*  EDIT THE          tcpindx   -  HIGH-LEVEL QUALIFIER FOR TCP LOAD DSN
//*                   indx      -  HIGH-LEVEL QUALIFIERS FOR EPS DSNS
//*                   volser1   -  VOLSER FOR PRIMARY CHECKPOINT DSN
//*                   volser2   -  VOLSER FOR BACKUP CHECKPOINT DSN
//*
//*  NOTE: SINCE EPS IS INSTALLED IN A SEPARATE TARGET
//*  LOAD LIBRARY FROM TCP, THE TCP LOAD LIBRARY
//*  MUST BE INCLUDED AS A STEPLIB DATASET IN ORDER
//*  MAKE THE SAS/C RUN-TIME LIBRARY AVAILABLE.
//*
//CKPTDEF  EXEC      PGM=IDCAMS
//SYSPRINT DD        SYSOUT=*,
//          DCB=BUFNO=1
//SYSIN    DD *

DELETE ( 'indx.CKPTNEW' )          CLUSTER
DELETE ( 'indx.CKPTNEW.BKUP' )    CLUSTER

DEFINE CLUSTER (
    NAME ( 'indx.CKPTNEW' )
    INDEXED
    KEYS ( 6 0 )
    SHAREOPTIONS ( 4 3 )
    VOLUMES ( volser1 )
    BUFFERSPACE ( 51200 )
    SPANNED
    UNIQUE )
DATA ( NAME ( 'indx.CKPTNEW.DATA' )
    CONTROLINTERVALSIZE ( 2048 )
    RECORDSIZE ( 512 32000 )
    CYLINDERS ( 5 5 )
    FREESPACE ( 20 20 )
    SPEED )
INDEX ( NAME ( 'indx.CKPTNEW.INDEX' ) )

DEFINE CLUSTER (
    NAME ( 'indx.CKPTNEW.BKUP' )
    INDEXED
    KEYS ( 6 0 )
    SHAREOPTIONS ( 4 3 )
    VOLUMES ( volser2 )
    BUFFERSPACE ( 51200 )
    SPANNED
    UNIQUE )
DATA ( NAME ( 'indx.CKPTNEW.BKUP.DATA' )
    CONTROLINTERVALSIZE ( 2048 )
    RECORDSIZE ( 512 32000 )
    CYLINDERS ( 5 5 )
    FREESPACE ( 20 20 )
    SPEED )
INDEX ( NAME ( 'indx.CKPTNEW.BKUP.INDEX' ) )
//*
//*  CHANGE THE CHECKPOINT FILE VERSION
//*
//CONVERT  EXEC      PGM=T07SCVCH PARM='-V'
//STEPLIB  DD        DISP=SHR,DSN=indx.T07LOAD
//          DD        DISP=SHR,DSN=tcpcindx.LOAD
//SYSPRINT DD        SYSOUT=*

```

```

//SYSTEM      DD      SYSOUT=*
//OLDCKPT     DD      DISP=SHR,DSN=indx.CKPT
//NEWCKPT     DD      DISP=OLD,DSN=indx.CKPTNEW
//*
//*  MAKE BACKUP COPY OF NEW CHECKPOINT FILE
//*
//REPRO       EXEC     PGM=IDCAMS,COND=(0,LT,CONVERT)
//SYSPRINT    DD      SYSOUT=*,
//            DCB=BUFNO=1
//SYSIN       DD      *

      REPRO      INDATASET( 'indx.CKPTNEW' )
                  OUTDATASET ( 'indx.CKPTNEW.BKUP' )
//*
//*  RENAME CHECKPOINT FILES
//*
//RENAME      EXEC     PGM=IDCAMS,COND=((0,LT,CONVERT),(0,LT,REPRO))
//SYSPRINT    DD      SYSOUT=*,
//            DCB=BUFNO=1
//SYSIN       DD      *

      ALTER      'indx.CKPT'
                  NEWNAME ( 'indx.CKPTOLD' )
      ALTER      'indx.CKPT.DATA'
                  NEWNAME ( 'indx.CKPTOLD.DATA' )
      ALTER      'indx.CKPT.INDEX'
                  NEWNAME ( 'indx.CKPTOLD.INDEX' )

      ALTER      'indx.CKPT.BKUP'
                  NEWNAME ( 'indx.CKPTOLD.BKUP' )
      ALTER      'indx.CKPT.BKUP.DATA'
                  NEWNAME ( 'indx.CKPTOLD.BKUP.DATA' )
      ALTER      'indx.CKPT.BKUP.INDEX'
                  NEWNAME ( 'indx.CKPTOLD.BKUP.INDEX' )
      ALTER      'indx.CKPTNEW'
                  NEWNAME ( 'indx.CKPT' )
      ALTER      'indx.CKPTNEW.DATA'
                  NEWNAME ( 'indx.CKPT.DATA' )
      ALTER      'indx.CKPTNEW.INDEX'
                  NEWNAME ( 'indx.CKPT.INDEX' )

      ALTER      'indx.CKPTNEW.BKUP'
                  NEWNAME ( 'indx.CKPT.BKUP' )
      ALTER      'indx.CKPTNEW.BKUP.DATA'
                  NEWNAME ( 'indx.CKPT.BKUP.DATA' )
      ALTER      'indx.CKPTNEW.BKUP.INDEX'
                  NEWNAME ( 'indx.CKPT.BKUP.INDEX' )

```

JCL Explanation

The sample T07SCVCH JCL, shown on the previous pages, defines a new set of checkpoint VSAM clusters, runs the checkpoint file conversion utility to fill the primary checkpoint file with converted records, copies the new primary checkpoint file to the new secondary checkpoint file, and renames the old and new checkpoint files.

Options to override T07SCVCH default behavior can be specified on a PARM argument on the EXEC statement. For more information, read [“Execution Options” on page 2-7](#).

In the conversion step, the SYSPRINT DD statement receives informational messages and the SYSTERM DD statement receives error messages. The OLDCKPT DD statement specifies the existing checkpoint file; the NEWCKPT DD statement specifies the file that is to be filled with converted records.

Execution Options

T07SCVCH can be run without specifying any options on the EXEC JCL statement. In that case, it converts the existing checkpoint file to the latest level.

To specify options to T07SCVCH, include the PARM argument on the EXEC statement. There are two supported options; one to specify the level to convert to and one to specify verbose mode operation. For example, to tell T07SCVCH to convert to level 2 and to display detailed messages (verbose), the EXEC statement would look like this:

```
//CONVERT EXEC PGM=T07SCVCH, PARM=' -S -v'
```

The dash before each option signals T07SCVCH that an option is starting (this is also a UNIX convention for specifying options to commands). The number 2 says to convert to level 2. The “v” directs T07SCVCH to operate in verbose mode (display detailed messages). The order of the arguments is unimportant. When duplicates of the same type of argument are present, the last one encountered (rightmost) is used.

Starting and Stopping SOLVE:EPS

You can start and stop SOLVE:EPS execution using operator commands.

Starting SOLVE:EPS

You can execute SOLVE:EPS as a started task or batch job.

Use this operator command to start SOLVE:EPS as a started task:

S *procedure_name*

procedure_name Name of the JCL procedure created in a PROCLIB known to JES2.

To run SOLVE:EPS as a batch job, execute the T07EPS JCL procedure, or include it in a batch job JCL stream. Submit the JCL from TSO or via a JES internal reader.

Stopping SOLVE:EPS

Use the operator command **SHUTDOWN** to terminate SOLVE:EPS. Read [“SHUTDOWN” on page 1-109](#) for a description of the **SHUTDOWN** command.

Using the **EPS SHUTDOWN** command is preferable to using the MVS **CANCEL** command, as the **CANCEL** command may not close the checkpoint and log files properly and there could be problems when you restart SOLVE:EPS.

Note: If SOLVE:TCPaccess is stopped, any BSD Inprocessors or BSD Outprocessors will terminate. Other SOLVE:EPS functions remain available. The BSD Inprocessors and BSD Outprocessors can be started when SOLVE:TCPaccess becomes available.

Recovery Action After Unexpected Termination

After SOLVE:EPS termination, whether normal or abnormal, simply restart SOLVE:EPS.

Even when abnormal termination occurs, no data is lost as SOLVE:EPS repeats any print job in progress when SOLVE:EPS terminated.

However, the SOLVE:EPS NJE Inprocessor runs as an NJE node. If SOLVE:EPS or the NJE Inprocessor terminates unexpectedly, JES2 determines that an NJE node is down. In this event, jobs being transmitted by JES2 to the SOLVE:EPS NJE node are placed on hold by the JES subsystem. To release a held job, issue the **\$A** command for the job. You are notified of this condition whenever JES issues a HASP532 message followed by JOB HELD.

SOLVE:EPS Commands Usage

You can control the SOLVE:EPS address space through the use of MVS **MODIFY (F)** commands entered from the console or from a TSO-based facility such as SDSF. Use the operator interface to perform these tasks:

- Obtain SOLVE:EPS status
- Control SOLVE:EPS tracing
- Dynamically control Inprocessors, Outprocessors, queues, and jobs
- Start and stop virtual printers

Status Display

Use the **SHOW** commands documented in the chapter [“SOLVE:EPS Operator Commands”](#) to obtain status about SOLVE:EPS, Inprocessors, Outprocessors, queues, or jobs:

F *name*,**SHOW** . . .

Trace Control

The tracing facility in SOLVE:EPS is used in the event that diagnostic data is required for problem determination and resolution. Normally all tracing is off.

Note: Call Customer Support if you need help using the TRACEWORD facility.

Tracing is done at the object (Inprocessor, Outprocessor, spool EPS) level. Each object has its own TRACEWORD that determines the type of tracing desired. The following table describes the TRACEWORD settings. The TRACEWORD settings can be combined by adding the TRACEWORD values. The trace output appears only in the SOLVE:EPS log.

Component	TRACEWORD Value	Description
TCP Alternate Console	1	Trace connection events.
	2	Trace connection stages (trace connection events must be set).
	4	Trace connection states (both of the above must be set).
	8	Trace socket I/O events.
	10	Trace socket I/O buffers (tracing socket I/O events must be set).
	20	Trace Work-To-Do (WTD) messages.
	40	Trace WTD message details (Trace WTD messages must be set).
	80	Trace low-level events.

TRACEWORD		
Component	Value	Description
e-Control Alternate Log	1	Trace Work-To-Do (WTD) messages.
	2	Trace WTD message details (trace WTD messages must be set).
	4	Trace PC call results.
	8	Trace PC call details (trace PC call results must be set).
	10	Write logged message to SYSOUT (CLASS=*).
	80	Trace low-level events.
BSD Inprocessor	1	Trace LPR protocol points.
	2	Trace buffer contents of network I/O.
	4	Trace intertask communication points.
BSD Outprocessor	1	Trace LPR protocol points.
	2	Trace buffer contents of network I/O.
	4	Trace intertask communication points.
	8	Trace converter result buffers.
	10	Trace state machine transitions.
	80	Trace converter state machine transitions.
DSO Outprocessor	1	Trace DSO processing flow points.
	2	Trace buffer contents of network I/O.
	4	Trace inter task communication points.
JES Outprocessor	1	Trace processing flow.
	2	Trace buffer contents of SPOOL output.
	4	Trace intertask communication points.
	8	Trace SYSOUT dynamic allocation text units.
	10	Trace the converter state machine.
NJE	1	Trace logic.
	2	Trace buffers.
	10	Trace buffer uncompressed.
NJE Outprocessor	1000	Trace flow.
	2000	Trace buffers.
	4000	Trace internal.
	8000	Trace compress buffers.
NJE Inprocessor	100	Trace logic.
VPT Inprocessor	1	Trace logic.
	2	Trace buffers.
SSI Inprocessor	1	Trace events.
	2	Trace interface.
All of the above	80000000	Turn off ABEND trap handling.

Dynamic Definition

The operator commands are used to dynamically define new objects within SOLVE:EPS, such as Inprocessors, Outprocessors, and queues.

Read the descriptions of the **DEFINE** and **SET** commands in the chapter [“SOLVE:EPS Operator Commands”](#) for more information.

Stopping and Restarting Virtual Printers

To start or stop an individual virtual printer, use the LOGICALUNITNAME parameter with the **START** or **STOP** command. See the descriptions of [“START” on page 1-110](#) and [“STOP” on page 1-111](#).

Using SOLVE:EPS-Administration

Instead of using the native command interface to operate SOLVE:EPS, the ISPF panel interface distributed with SOLVE:EPS is a more friendly approach. See the [“Using SOLVE:EPS-Administration”](#) appendix for details.

Message and Log Controls

SOLVE:EPS produces messages that will be written to the MVS operator console and logged to a SOLVE:EPS designated log. These are the types of messages there are:

- Error messages
- Informational messages
- Response messages
- Tracing messages

All messages except response messages are controllable via **SET** commands for each object. For more information, read the description of the **SET** commands in the [“SOLVE:EPS Operation”](#) chapter.

Error Messages

There are three types of error messages: FATAL, ERROR, and WARNING.

- A FATAL message means that a catastrophic error has occurred and part, or all, of SOLVE:EPS cannot continue.
- An ERROR message indicates that some kind of error occurred and action should be taken to correct it.
- A WARNING message indicates that an action did not happen entirely correctly, but SOLVE:EPS could make adjustments or use defaults to correct itself.

Informational Messages

There are two types of informational messages: INFO and DEBUG.

- INFO messages are produced by SOLVE:EPS to inform the operator about SOLVE:EPS operations and actions in progress.
- DEBUG messages are produced by SOLVE:EPS to help diagnose problems or provide further detailed information about SOLVE:EPS operations and actions.

Response Messages

Response messages are those messages that SOLVE:EPS always produces in response to a command like **SHOW** or **SET**. The messages always appear so the operator knows that SOLVE:EPS has recognized the command and taken action.

In the case of a **SHOW** command, the response message is the results of the **SHOW** command. Other commands respond with a success or failure message.

Trace Messages

Trace messages appear only in the log. These messages trace events through SOLVE:EPS and should be used only at the direction of Customer Support as the amount of messages generated by a trace can be quite large.

3

Using SOLVE:EPS

This chapter describes how to use the SOLVE:EPS (Enterprise Print Services) software once it has been installed and properly configured. It includes these sections:

- [“Printing on Remote Printers” on page 3-2](#)

Describes how to use the JES2/NJE interface and the virtual printer emulator.

- [“Printing on JES Printers” on page 3-5](#)

Describes how to use the UNIX **lpr** command and the supported **lpr** command options. Also provides examples of **lpr** usage.

Printing on Remote Printers

This section describes how to direct SYSOUT data sets and SNA application output via SOLVE:EPS to remote printers on the TCP/IP network. These outbound print jobs are processed by the SOLVE:EPS BSD Inprocessor and the SOLVE:EPS virtual printer.

Note: For DBCS data coming from or going to JES, the default handling of SO/SI occupies space. To override this behavior, the job must have PRMODE set to SOSI2. On the mainframe, an OUTPUT statement is necessary to specify PRMODE. For requests inbound to the mainframe, the user can specify PRMODE on the extended class, or the SOLVE:EPS administrator can specify it in the BSD Security Table.

Using the JES2/NJE Interface

Use standard JCL to route outbound print data via JES/NJE and SOLVE:EPS to remote LPD servers or to LAN-connected printers.

With the JCL statement below output print data is passed to a SOLVE:EPS BSD Outprocessor and on to a remote LPD server, or to a DSO Outprocessor and on to a LAN-connected printer, on the TCP/IP network:

```
//dd_name DD SYSOUT=A,DEST=( epsnode,epsdest )
```

<i>epsnode</i>	SOLVE:EPS NJE node name defined to JES.
<i>epsdest</i>	Destination defined to the SOLVE:EPS application with the DEFINE QUEUE parameter, which maps to the desired remote host IP address and printer name.

Dynamic allocation can also specify the node and destination pair, letting TSO applications access a BSD or DSO Outprocessor with commands such as PRINTDS.

When the Outprocessor type is BSD, JES2 printing options are passed to the remote print server. The SOLVE:EPS BSD Outprocessor maps the JES2 options into the C-record (extended class parameter) of the LPD control file.

For NJE and SSI Inprocessors, you can control output JES-to-ASCII conversion by setting the UCS for JES spool data sets. You must set the UCSTABLE parameter of the Inprocessor and make sure the UCS setting matches the desired table entry. Read the descriptions for the **SET INPROCESSOR** parameter [“UCSTABLE” on page 1-44](#), and for the section “UCS Table” in “SOLVE:EPS Members” (Chapter 8) of the *SOLVE:EPS Installation and Administration Guide*.

PRINTDS

This is an example of using the TSO **PRINTDS** command to route a print file via SOLVE:EPS to a printer on the TCP/IP network:

PRINTDS DA(dataset) DEST(epsnode.epsdest)

<i>dataset</i>	MVS data set to be printed.
<i>epsnode</i>	SOLVE:EPS node name defined to JES.
<i>epsdest</i>	Remote printer destination name as defined in the SOLVE:EPS DEFINE QUEUE configuration statement that maps the name to its remote printer.

Note: Use the TSO **HELP PRINTDS** command for additional help with this command.

Using the Virtual Printer Emulator

SOLVE:EPS provides a facility for applications to utilize remote printers attached to a TCP/IP network. Refer to the documentation for these applications, such as CICS, IMS, or ROSCOE, that print to virtual printers.

SOLVE:EPS virtual printer names are defined by APPL statements in the VTAMLST definition and the VPRTR Inprocessor LOGICALUNITTABLE T07VPT T07PARM member. The LOGICALUNITTABLE member contains the VTAM luname and its associated queue for printing to a remote host and remote printer. When an SNA printer application requests a SOLVE:EPS virtual printer, print data is routed via SOLVE:EPS to the corresponding remote host and print queue name.

Read “Configuring VTAM” (Chapter 6) in the ***SOLVE:EPS Installation and Administration Guide*** for more information on the VTAM definition required to set up virtual printers.

Using LU3

SOLVE:EPS provides LU3 functionality that lets users print to SNA LU3 printers.

LU3 Limitations

SOLVE:EPS LU3 processing includes these limitations:

- Processing simulates buffered printer operations, which causes these actions:
 - Attribute characters, and alphanumeric characters in non-display/non-print fields are transferred as spaces. When these characters constitute an entire line, the line prints as spaces and a blank line appears in the printout.
 - The NL order executes only during an unformatted printout.
- Data is always processed in mixed case. Whether or not the output is displayed in mixed case or upper case only depends on the physical characteristics of the printer's case switch setting.
- Processing transfers form feeds (FF) to output print when they occur in the input stream immediately following a write control character (WCC) or when they are placed in column one (1) of a print row (line). NL and CR orders are only valid in unformatted data streams. Whether or not actual forms skipping occurs in the printed output depends on physical printer switch settings.
- SOLVE:EPS is unable to detect a 3270 data stream when the DLOGMOD(BIND) in the VTAM APPL printer definition specifies an LU1 session. SOLVE:EPS attempts to process the stream as an LU1 SCS data stream, producing unpredictable results.

Note: These notes apply:

Nulls are translated to blanks.

With few exceptions, unprintable characters are translated to blanks.

Write Structured Field and Structured Field functions are not supported.

High-intensity attribute is not supported. Fields that include this attribute are printed with normal intensity.

Printing on JES Printers

This section describes how to direct print files from remote hosts on the TCP/IP network to MVS for printing.

Note: For DBCS data coming from or going to JES, the default handling of SO/SI occupies space. To override this behavior, the job must have PRMODE set to SOSI2. On the mainframe, an OUTPUT statement is necessary to specify PRMODE. For requests inbound to the mainframe, the user can specify PRMODE on the extended class, or the SOLVE:EPS administrator can specify it in the BSD Security Table.

Using the UNIX lpr Command

From a remote UNIX host with Berkeley-based UNIX, use the **lpr** command to direct print files via SOLVE:EPS to JES for processing. For this use of the **lpr** command, in addition to names of files to be printed, SOLVE:EPS requires that -P and -C options specify the SOLVE:EPS print queue being used and the extended class parameters that convey printing characteristics to JES.

**lpr -Pprinter -Cclass -#copies -Jjob -Ttitle -imargin -xfont
-wcols -r -m -h -s -yfilter filename filename...**

<i>printer</i>	Printer or print queue name.
<i>class</i>	Print class or extended class parameters.
<i>copies</i>	Number of copies to print.
<i>job</i>	Job name to appear on the banner page.
<i>title</i>	Title, instead of filename, for banner page.
<i>margin</i>	Number of columns to indent the left margin.
<i>font</i>	Font and font position assignment.
<i>cols</i>	Page width.
-r	Remove file after printing.
-m	Send notification by mail after printing.
-h	Suppress banner page.
-s	Create symbolic link for print file(s).
<i>filter</i>	Filter for processing special types of print files.

The files to print become LPD data files and the arguments become the LPD control file when the print job is sent to SOLVE:EPS.

The -P and -C arguments are described in detail in the following topics. The remaining arguments are described in the ***Sun OS Reference Manual*** and other UNIX manuals.

The -P Option

The `-P` option defines a printer.

The printer must be defined in the UNIX system file `/etc/printcap` as a SOLVE:EPS printer.

- The remote machine (rm=) must be the host name for the SOLVE:TCPaccess used by SOLVE:EPS.
- The remote printer (rp=) must be defined in the BSD security table or as a SOLVE:EPS DEFINE QUEUE configuration statement.

Here is a sample `/etc/printcap` file:

[illegible]

The -C Option

The -C option in the extended class parameter communicates all necessary JES parameters.

In the UNIX environment it is basically a free-format field. Here it is given a set of arguments defining the JES option for processing the print file.

This is the general format of a -C option:

-Cc:rm:key=value:key=value...

This option includes two positional arguments, specifying the SYSOUT class and room number.

- Each keyword parameter has the form key=value.
- Individual parameters are separated by a colon (:).
- When the SYSOUT class is omitted, the room number must be preceded by a colon (:).
- If a value consists of several items, they are separated by commas. For example, :ch=A0A,B0A,FM15,GT12.

For details of each parameter, read the section [“Supported LPD Command Options” on page 3-8](#), following.

This example routes a print file to class A and assigns it to room 1265:

-CA:1265

Supported LPD Command Options

This table lists the supported options in the extended class parameter.

- The **Keyword** column lists keywords as coded in the -C option on the **lpr** command.
- The keywords in the MVS column are as generated for the OUTPUT statement when the SOLVE:EPS LPD server passes the print file to JES.
- If the **X** column contains an N, the data is not translated to uppercase; otherwise, data may be entered in upper or lowercase and is translated as necessary.

Keywords can be entered in upper or lowercase.

KEYWORD	MVS	X	COMMENTS
-	CLASS		SYSOUT class
-	ROOM		Room. 1-4 characters.
ACCOUNT	N/A	N	Sets the account field (NJHGACCT).
CH	CHARS		Character arrangement tables for 3800 printing subsystem. May have up to four elements as: :CH=TBL1,TBL2,TBL3,TBL4: If any elements are coded then the other elements are cleared to their default value.
CL	CLASS		An alternate way to provide SYSOUT class.
D	DEST		Coded as node.userid or node.Rnnn. node is moved to NDHGNODE and is a method of sending files beyond JES. userid or Rnnn is moved to NDHGRMT (see \$NHD in SYS1.HASPSRC) and is meaningful to the receiving system. Use JES2 exit (EXIT 13) to determine whether the value in NDHGRMT is a legitimate JES2 DESTID or other routing and not a TSO user. Similarly, it may be necessary to use remote routing as JES2 does not support NJE routing to a special local printer.
F	FORMS		Special forms requested.
FC	FCB		Forms Control Buffer
FORMDEF	FORMDEF		Form Definition (see an applicable JCL reference manual for valid values.)
IN	N/A		Indent.
JOBIN	N/A		Job ID.
JOBN	N/A		Set the job name in job header (NJHGJNAM).
LC	LINECT		Lines per page. Numeric: 0<=value<=255.
PAGEDEF	PAGEDEF		Page definition (see an applicable JCL reference manual for valid values.)
PGMR	N/A	N	Set programmer name in job header (NJHGPRGN).
PRIORITY	N/A		SOLVE:EPS job priority.

KEYWORD	MVS	X	COMMENTS
PRMODE	PRMODE		Processing Mode. One of the following: • LINE - Sends data set to a line mode printer • PAGE - Sends the data set to a page mode printer
R	N/A		An alternate way to provide the JES2 ROOM.
TR	N/A		Specifies the language translation to apply to the individual print files, normally ASCII-to-EBCDIC translation. If not specified, the translation selected by the LPD server security table is used. If no translation is given there, then the translation of the BSD Inprocessor is used, if present. A job that has no translation specification will use the default language translation as determined by the translate table member of the PARMLIB DD.
UC	UCS		• UCS=##NC Indicates that the print data is not to be translated or transformed in any way. • UCS=# Suppresses the carriage control processing. The file types that cause this include raster images or DVI files. It is assumed that a "filter" program is provided to handle these.
USER	NOTIFY	N	Set NJHGUSID.
WI	N/A		Maximum page width or line length including JES carriage control character. Numeric: 0<=value<=65535. Zero causes a default value to be used. If a CR, LF or FF is not encountered within this number of characters then a new line is forced. NOTE: value-1 is the maximum length for data.
WRITER	WRITER		Name of external writer to be passed to JES.

Example of lpr Usage

UNIX clients can access printers attached to the mainframe host by routing print data with the UNIX **lpr** command. These examples are typical entries to route print from the UNIX systems to the MVS host:

lpr -Pr20 -Ca:user=*myname*:d=jesa.rmt20 *my.file*

In this example, UNIX file *my.file* is to be printed on r20 according to JES processing options in the -C extended class parameter. Print queue r20 should be defined on the user's UNIX system as a SOLVE:EPS print queue, which, in turn, is defined in the SOLVE:EPS configuration file. The SYSOUT class is A. The JES NJE control record field NJHGUSID is assigned the value *myname*. The UNIX system sends the print job to SOLVE:EPS on the MVS system, and SOLVE:EPS sends the print job to JES on that MVS system, but the ultimate destination is remote printer rmt20 on a JES system called JESA.

lpr -Plp -C:d=jesa.local:prmode=page:ch=fm15,gt15:pagedef=08081:wi=75 *my.3800.file*

In this example, UNIX file *my.3800.file* is to be printed on lp, which should be defined suitably in both the UNIX and SOLVE:EPS printer configurations. The SYSOUT class is A by default. The ultimate destination is the local JES printer on JES system JESA. The processing mode is PAGE with 3800 printer character arrangement tables fm15 and gt15. Print width is 75 (carriage control plus 74 data positions per line).

A

User Exits

This appendix describes the user exits. It includes these sections:

- [“Overview” on page A-2](#)
Provides a brief overview of what the SOLVE:EPS user exits can be used for.
- [“User Exit 1 – NJE Header/Trailer” on page A-3](#)
Describes User Exit 1, which pertains to NJE Header/Trailer Record Modification.
- [“User Exit 2 - Control File Line” on page A-4](#)
Describes User Exit 2, which pertains to BSD control file line modification.
- [“Dispatch User Exit” on page A-6](#)
Describes the Dispatch User Exit, which lets you change queue and/or job settings before the job is given to the Outprocessor.
- [“Dataset User Exit” on page A-15](#)
Describes the Dataset User Exit, which lets you insert information to send to the print destination.
- [“CONVERTSELECTOR Outprocessor User Exit” on page A-32](#)
The CONVERTSELECTOR user exit, assigned to an Outprocessor, lets you override the SOLVE:EPS converter selection.
- [“Initialization User Exits” on page A-34](#)
Describes Initialization user exits, which set the INITEXIT parameter on an existing converter and define a new converter that has INITEXIT specified.
- [“Command Security User Exit” on page A-47](#)
Describes the command security user exit, which authorizes use of SOLVE:EPS commands, particularly the SOLVE:EPS-Administration (SOLVE:EPS-Admin) interface.
- [“Response User Exit” on page A-53](#)
Describes the response user exit, which prevents the display of selected messages.
- [“Shared Fields Tables” on page A-54](#)
Provides tables describing object fields that are common to the commands and other tables described in this document.
- [“Overriding Character Handling for LU1” on page A-58](#)
Describes how to override the default SOLVE:EPS character handling for LU1 conversion.

Overview

User exits within SOLVE:EPS are provided for the user to change the default behavior of SOLVE:EPS. Exits occur at certain key points within SOLVE:EPS processing in order for modifications or process decisions to occur.

The exits vary in function and vary in the information accessible from SOLVE:EPS, depending on function.

This information is potentially available to exits:

- The Job Control Block (JCB) contains all data relating to a job and is typically displayed with a **SHOW JOB** command. The macro for this exit is JCB\$.
- The Queue Control Block (QCB) contains all data relating to a queue and is typically displayed with a **SHOW QUEUE** command. The macro for this exit is QCB\$.
- The SOLVE:EPS Data set Control Block (EDCB) contains data relating to a spool (temporary) file within a job. There is one or more of these per job. The macro for this exit is EDCB\$.

All macros for this facility are located in the T07MAC library.

Exits are typically written in Assembler language and follow the MVS register standard, which is:

- | | |
|-----|----------------------------------|
| R1 | Pointer to the parameter list. |
| R13 | Pointer to a register save area. |
| R14 | Return address. |
| R15 | Entry point. |

All other registers have unknown contents.

To assemble and link a user exit, see the sample JCL named T07XXASM in the T07SAMP library.

User Exit 1 – NJE Header/Trailer

User Exit 1 pertains to NJE Header/Trailer Record Modification. This exit is used to analyze and possibly modify NJE header/trailer records created by SOLVE:EPS before they are sent to the NJE partner (typically JES). The exit can add sections to the header, if desired, as long as the length field provided in the parameter list is updated. The total header size must be no larger than 32K. The exit is called once for each type of NJE header (job header, data set header, and job trailer).

The sample code is in T07SAMP member T07XX001.

The load module for User Exit 1 is T07XX001. Parameters for User Exit 1 are as follows:

Table A-1 User Exit 1 Parameters

Parameter List Offset	Field	Description
(0)	Header type	Address of a fullword value containing the header type. The values are: 1 - Job header 2 - Data set header 3 - Job trailer
(4)	NJE Buffer	Address of the NJE record indicated by the header type that SOLVE:EPS has built. This buffer is 32K and modifications occur within the buffer. If modifications occur outside the buffer, unpredictable results may occur.
(8)	NJE Header Length	Address of a fullword containing the current length of the NJE record indicated by the header type. If modifications occur to the NJE record causing an increase or decrease in length, this field must be updated with the new length.
(C)	JCB (Job Control Block)	Address of the JCB SOLVE:EPS used to create the NJE record. This control block is read only.
(10)	QCB (Queue Control Block)	Address of the QCB SOLVE:EPS used to create the NJE record. This control block is read only.
(14)	EDCB (SOLVE:EPS Data set Control Block)	Address of the EDCB used to create the NJE data set header record. This parameter is present only for the data set header type. This control block is read only.

User Exit 2 - Control File Line

User Exit 2 pertains to BSD control file line modification. Most of the control file lines are passed to this exit for inspection and modification before being sent to the remote host. This exit can change the values SOLVE:EPS supplies, completely replace the line with its own data, or ask SOLVE:EPS to omit the line.

SOLVE:EPS calls User Exit 2 once for each of the following control file lines (not necessarily in this order):

C	Class for banner page
H	Host name
J	Job name for banner page
L	Print banner page
M	Mail when printed
P	User identification
T	Title for pr filter

SOLVE:EPS lets the exit omit only line types J, M, and T. SOLVE:EPS sends line M only if there is data in the buffer returned by User Exit 2; SOLVE:EPS calls the exit for line T only if the filter setting on the queue is p (this filter instructs the remote line printer daemon to pipe the data through the pr filter before printing). If the queue has NOBANNER set, SOLVE:EPS does not call User Exit 2 for the L line. Conversely, if the queue has BANNER set, SOLVE:EPS will always send an L line.

There is a sample exit in source code form in T07SAMP member T07XX002.

Table A-2 User Exit 2 Parameters

Parameter List Offset (hex)	Field	Description
0	Control file line type	This parameter is passed as an integer, but the type is actually in the low-order byte as the EBCDIC value for the control file line. For example, for the C control file line, this parameter would be 0x000000C3.
4	Control file line buffer	This parameter is a pointer to the line buffer that User Exit 2 can use. On entry, it is filled in by SOLVE:EPS with data that would be sent if User Exit 2 were not used.
8	Control file line buffer maximum length	This parameter is an integer that expresses the size of the control file line buffer. It does not specify the length of the data that SOLVE:EPS has filled in.
C	Job Control Block (JCB)	This parameter is a pointer to the JCB for the SOLVE:EPS job. Normally, SOLVE:EPS uses fields from this control block to build control file lines. It can be mapped in an assembler program by the JCB\$ member of T07MAC.

Parameter List Offset (hex)	Field	Description
10	SOLVE:EPS queue name	This parameter is a pointer to a string representing the name of the SOLVE:EPS queue that dispatched this print job.
14	Remote host name	This parameter is a pointer to a string representing the name of the remote host to which this print job is being sent.
18	Remote printer name	This parameter is a pointer to a string representing the name of the printer on the remote host to which this print job is being sent.

All strings in the parameter list conform to the C language (they are null terminated). If User Exit 2 fills in the control file line buffer, it must ensure that the data is null terminated. It must not append any line termination characters to the data, since SOLVE:EPS does that.

When User Exit 2 completes, it must pass a return code back to SOLVE:EPS indicating how it wants the control file line to be handled. The following table describes the valid return code values:

Table A-3 Valid Return Codes for User Exit 2

Return Code Value (hex)	Description
0	SOLVE:EPS is to use the control file line it supplied. User Exit 2 is not overriding the line.
4	User Exit 2 has supplied an override control file line in the buffer provided to the exit. The buffer contents are in EBCDIC, so SOLVE:EPS is to translate them to ASCII before sending them to the remote system.
8	User Exit 2 has supplied an override control file line in the buffer provided to the exit. The buffer contents are already in ASCII, so SOLVE:EPS is not to translate them before sending them to the remote system.
C	User Exit 2 wants SOLVE:EPS not to send the control file line to the remote system. SOLVE:EPS may ignore this request. For example, if User Exit 2 returns this code for an H line, SOLVE:EPS will send the line anyway, since it is required by protocol.

Dispatch User Exit

The Dispatch User Exit is called by SOLVE:EPS just before a job is dispatched from a queue to an Outprocessor. It lets you change queue and/or job settings before the job is given to the Outprocessor. It may also alter the dispatching process itself by positioning dispatch, or by requeuing, holding, or deleting the job. It is written in the new User Exit Language, EPSUSEX. The EPSUSEX user exit language is described in the [“EPSUSEX User Exit Language”](#) appendix.

Specifying the Dispatch User Exit

The Dispatch User Exit is specified on the queue(s) for which it is to be called. Multiple queues may use the same exit, and multiple exits may be used across a SOLVE:EPS system.

The DISPATCHEXIT keyword assigns a Dispatch User Exit to a queue:

SET QUEUE *q_name* **DISPATCHEXIT** *member*

Syntax Description

<i>q_name</i>	Specifies the queue to which the exit is being assigned.
<i>member</i>	Specifies the member in PARMLIB DD containing the EPSUSEX code for the exit.

Read the description of the command [“SET QUEUE” on page 1-62](#) for more information.

The Dispatch User Exit code is loaded when the queue is started and it is called before the dispatch of every job. It may be refreshed without stopping the queue:

REFRESH DISPATCHEXIT *member*

Syntax Description

<i>member</i>	Specifies the member in the PARMLIB DD to refresh.
---------------	--

Read the description of the command [“REFRESH” on page 1-14](#) for more information.

Data Objects Available to the Dispatch User Exit

The following is a list of the objects that SOLVE:EPS makes available to the Dispatch User Exit. The object's fields are explained in the following sections.

This table describes the SOLVE:EPS Dispatch User Exit objects:

Table A-4 Dispatch User Exit Objects

Object	Description
queue	Selected fields from a copy of the queue data structure. Changes made to the structure are only in effect for the execution of a single SOLVE:EPS job.
job	Selected fields from the data structure for a SOLVE:EPS job. Changes made to the structure stay with the job until they are changed again, or until the job is purged.
boolean	A constant object holding true and false values for testing of, and assignment to, boolean-type numeric fields.
rc	A single field Dispatch User Exit return code. Use the return object to set its value.
return	A constant object containing fields for setting the rc object.
message	A single-field object for returning a message from the exit. If a value is set, SOLVE:EPS displays it (T07MQ049I).

queue Object Fields

Selected fields from a copy of the queue data structure are made available to the Dispatch User Exit. Changes made to the structure are only in effect for the execution of a single SOLVE:EPS job.

Table A-5 Queue Object Fields

Field Name	Data Type	Modifiable	Comments
description	string	No	This field is not used by SOLVE:EPS directly; it is a field that customers can assign to suit their own purposes. It is set by the command SET QUEUE <i>q_name</i> DESCRIPTION <i>desc</i> .
nobanner	number	Yes	Determines whether a print request to a BSD lpd server suppresses the banner page control file line (the L line). If this field is set, it should be done by referencing the boolean object. For example, <code>queue nobanner := boolean true</code>
outpname	string	No	The name of the Outprocessor used by the queue.
prefixmember	string	Yes	This field names a member of the PARMLIB DD that is assigned by the command SET QUEUE <i>q_name</i> CONTROLSTRINGPREFIX MEMBER <i>member</i> . If the Dispatch User Exit sets this field, and the member name is different from that specified in the original queue, the member is loaded and used for the print job.
suffixmember	string	Yes	This field names a member of the PARMLIB DD that is assigned by the command SET QUEUE <i>q_name</i> CONTROLSTRINGSUFFIX MEMBER <i>member</i> . If the Dispatch User Exit sets this field, and the member name is different from that specified in the original queue, the member is loaded and used for the print job.

job Object Fields

Selected fields from the data structure for a SOLVE:EPS job are made available to the Dispatch User Exit. Changes made to the structure stay with the job until they are changed again, or until the job is purged.

Table A-6 Job Object Fields

Field Name	Data Type	Modifiable	Comments
accounting	string	Yes	Job accounting information
char1	string	Yes	Character arrangement table 1. If set, job trc should be turned on.
char2	string	Yes	Character arrangement table 2. If set, job trc should be turned on.
char3	string	Yes	Character arrangement table 3. If set, job trc should be turned on.
char4	string	Yes	Character arrangement table 4. If set, job trc should be turned on.
class	string	Yes	Job class.
node	string	Yes	Destination node name.
dest	string	Yes	Destination user name.
fcb	string	Yes	FCB.
flash	string	Yes	Flash forms overlay.
formdef	string	Yes	PSF FORMDEF.
forms	string	Yes	FORMS.
inpname	string	No	Name of the Inprocessor that created this job.
inptype	string	No	4-character type of the Inprocessor that created this job.
jobname	string	No	MVS job name of the originator of the print request.
linecount	number	No	Lines per page.
originnode	string	No	Origin node.
originuser	string	No	Origin user.
pagedef	string	Yes	PSF PAGEDEF.
priority	number	Yes	SOLVE:EPS job priority (1-999). To change the dispatching priority, set this field and then set the return code asking that the job not be dispatched now. For example: job priority := 800. rc value := return dontdispatch.
prmode	string	Yes	Processing mode.
programmer	string	No	Programmer name.

queueName	string	Yes	Name of the queue from which this job is being dispatched. To requeue the job (and not print it now), change this field to the desired queue, and set the return code not to bypass dispatch, or to hold the job: job queueName := 'OTHER'. rc value := return dontdispatch. // OR rc value := return hold.
room	string	No	Specifies the room number for the job.
trc	bit	Yes	Specifies that the data contains a Table Reference Character.
ucs	string	Yes	UCS (Universal Character Set).
spoolbytes	number	No	Number of bytes this job occupies in SOLVE:EPS spool; in other words, in the temporary data set.
completions	number	No	Count of the number of times this job has been sent successfully.
resets	number	No	Count of the number of times this job has been RESET by the operator.
failures	number	No	Count of the number of times this job has been sent unsuccessfully.

boolean Object Fields

A boolean object field is a constant object holding true and false values for testing of, and assignment to, boolean-type numeric fields.

Table A-7 Boolean Object Fields

Field Name	Data Type	Modifiable	Comments
true	number	No	A boolean true value (1).
false	number	No	A boolean false value (0).

rc Object Fields

An rc object field is a single field Dispatch User Exit return code which controls the dispatching process. Use the return object to set its value.

Table A-8 RC Object Fields

Field Name	Data Type	Modifiable	Comments
value	number	Yes	Return code for the Dispatch User Exit. Its initial value is return normal (possible values are shown in the following return fields table). It should be assigned values only from the return object's fields.

return Object Fields

A return object field is a constant object containing fields for setting the rc object.

Table A-9 Return Object Fields

Field Name	Data Type	Modifiable	Comments
normal	number	No	Return code that indicates that SOLVE:EPS should go ahead and dispatch the job, using whatever data structure modifications the Dispatch User Exit has made.
dontdispatch	number	No	Return code that indicates that SOLVE:EPS should not dispatch the job now. If the job queue name field has changed, the job will be requeued to that queue (it could go to the orphan queue if an unknown queue name is set). Whether the queue name changes or not, the queue is requested to re-dispatch.
hold	number	No	Return code that indicates that SOLVE:EPS should not dispatch the job, and should place it on hold status. If the job queue name field has changed, the job will be requeued to that queue (it could go to the orphan queue if an unknown queue name is set).
delete	number	No	Return code that indicates that SOLVE:EPS should not dispatch the job, and should delete it.

message Object Fields

The message object field is a single-field object for returning a message from the exit.

Table A-10 Message Object Fields

Field Name	Data Type	Modifiable	Comments
text	string	Yes	Message string that the Dispatch User Exit can set to explain its actions. Its initial value is an empty string. If the exit does not assign a value, no message is issued; if it does assign a value, message T07MQ049I is issued.

Sample Dispatch User Exit

A sample Dispatch User Exit is distributed as the T07XX003 member of the PARMLIB DD.

Tips

This section shows some common tasks that a Dispatch User Exit might carry out along with sample code.

Changing Control Strings

When using a BSD Outprocessor, it is possible to send prefix and suffix control strings that might be used to control a printer's settings. Normally they are associated with a SOLVE:EPS queue, but by using the Dispatch User Exit, these control strings can be specified based on a customer-determined combination of queue and job settings.

For this example, we assume that the queue description is set in the SOLVE:EPS configuration file to describe the type of printer associated with the queue. The job's FCB or FORMS attribute is used to determine whether landscape printing is desired.

```
if( queue description (==) 'hp_pcl' AND
    (job fcb (==) 'land' OR job fcb (==) 'wide' OR
    job forms (==) 'land' OR job forms (==) 'wide') )
{
    queue prefixmember := 'T07HPJL'.
    queue prefixmember := 'T07HPJX'.
}
```

Requeuing a Job

If during execution of the Dispatch User Exit it is determined that a job should not be dispatched from the queue it is on, but from another queue, it is possible to direct SOLVE:EPS to requeue it. To do this, change the job queue name and set the Dispatch User Exit return code to one of the values that prevents the job from being dispatched, either return dontdispatch or return hold:

```
if( job spoolbytes > 100000 AND job queue name (!=) 'HIGHCAPACITY' )
{
    job queue name := 'HIGHCAPACITY'.
    rc value := return dontdispatch.
    break.
}
```

In this code segment, if a job has more than 100,000 bytes in SOLVE:EPS spool and is not already on the HIGHCAPACITY queue, it is not dispatched. In addition, it is requeued to the HIGHCAPACITY queue and immediately dispatched from there (subject to queue state, session count limitations, etc.).

If the return code is set to return hold, instead of return dontdispatch, the job is placed on the HIGHCAPACITY queue in hold status. Operator intervention would be necessary to release it.

The break statement at the end of the statement block stops Dispatch User Exit execution immediately. Any following statements in the exit are bypassed.

Holding a Job

A variation of the above example would be to not dispatch a job based on some criteria, and to have it held.

```
if( job spoolbytes > 100000 AND job ucs (!=) 'OVER' )
{
    job ucs := 'OVER'.
    rc value := return hold.
    break.
}
```

In this example, if a job exceeds 100,000 bytes in SOLVE:EPS spool, and does not have a UCS setting of OVER, it is placed on hold status. (The queue name is not changed, so the job would remain on the same queue.) Pre-emptively, UCS is set to OVER, so that by just releasing the job, the operator can send it to its destination.

The break statement at the end of the statement block prevents subsequent exit statements from executing.

Changing a Job's Dispatching Priority

It is possible to prioritize SOLVE:EPS dispatching based on job size by using the Dispatch User Exit. In the following example, jobs over a certain size will have their priority adjusted, and they will be redispached.

```
if( job spoolbytes >= 10000 AND job priority <= 500 )
{
  if( job spoolbytes < 20000 ) { job priority := 600. }
  elseif( job spoolbytes < 50000 ) { job priority := 700. }
  elseif( job spoolbytes < 75000 ) { job priority := 800. }
  else { job priority := 900. }
  rc value := return dontdispatch.
  break.
}
```

SOLVE:EPS dispatching works by collecting all eligible jobs for a queue or queues, sorting them by priority and creation time, and then assigning them, one at a time, to an Outprocessor (subject to queue and Outprocessor session count restraints). The Dispatch User Exit is called once for each assignment. Thus for one dispatch cycle, if the Dispatch User Exit asks that a job not be dispatched now, SOLVE:EPS remembers the queue and moves on to the next job. When the cycle is complete, it redispaches for each queue that had a postponed dispatch. What this means is that by changing job priority and postponing dispatch, jobs can effectively be reprioritized, and dispatched accordingly.

By default, SOLVE:EPS sets a job's priority to 500 when it creates it, and the above code uses that fact to decide if it has already set a priority. It also checks the SOLVE:EPS spool byte count to see if it is over 10,000, sets the priority according to size, and asks SOLVE:EPS not to dispatch the job at this time. The net effect is that larger print jobs print after shorter ones.

Dataset User Exit

The Dataset User Exit, EPSUSEX, is called by SOLVE:EPS before and after conversion of each data set of a SOLVE:EPS job. It can insert information to send to the print destination. The information could be a banner page, printer control strings, or a combination of both.

The EPSUSEX user exit language is described in the [“EPSUSEX User Exit Language”](#) appendix.

Specifying the Dataset User Exit

The Dataset User Exit is specified on the queue(s) for which it is to be called. Multiple queues may use the same exit, and multiple exits may be used across a SOLVE:EPS system.

There are two available flavors of the Dataset User Exit, Assembler and EPSUSEX, described following. Only one type of exit can be assigned to a queue at any given time.

Assembler Dataset User Exits

An Assembler exit must be assembled and link-edited into the SOLVE:EPS load library concatenation. It is specified on the queue with the DATASETEXIT keyword:

SET QUEUE *q_name* DATASETEXIT MODULE *mod_name*

Syntax Description

q_name Specifies the queue to which the exit is being assigned.

mod_name Specifies the executable member of the SOLVE:EPS load library concatenation.

See the description of the command [“SET QUEUE” on page 1-62](#) for more details.

EPSUSEX Dataset User Exits

An EPSUSEX Dataset User Exit is loaded from the PARMLIB DD and compiled when the first queue referencing it is started.

Note: Compilation errors will prevent the queue from starting.

Use the keyword DATASETEXIT to assign a Dataset User Exit to a queue.

SET QUEUE *q_name* **DATASETEXIT MEMBER** *mem_name*

Syntax Description

<i>q_name</i>	Specifies the queue to which the exit is being assigned.
<i>mem_name</i>	Specifies the member in PARMLIB DD containing the EPSUSEX code for the exit.

It is possible to refresh an EPSUSEX Dataset User Exit.

REFRESH DATASETEXIT *mem_name*

Syntax Description

<i>mem_name</i>	Specifies the member in the PARMLIB DD to refresh.
-----------------	--

See the description of the command [“SET QUEUE” on page 1-62](#) for more details.

Assembler Interface

When calling a Dataset User Exit that has been written in Assembler, SOLVE:EPS uses standard MVS linkage conventions.

SOLVE:EPS Dataset User Exit Assembler Interface

The Assembler macros shown in the following table are distributed with SOLVE:EPS and must be copied into the Dataset User Exit source file (Assembler **COPY** command):

Table A-11 Assembler Interface Macros

Macro Name	Description
QCB\$	Maps the Queue Control Block.
JCB\$	Maps the Job Control Block.
EDCB\$	Maps the SOLVE:EPS Dataset Control Block.
OXDSCAL\$	Defines constants for the return code, call type, and return string options.

Entry

An Assembler Dataset User Exit is entered with standard MVS linkage. In the following table describing the Assembler Dataset User Exit parameter list, Register 1 points to a parameter list (the VL-bit for the last entry is not set):

Table A-12 Assembler Dataset User Exit Parameter List

OXDSCAL\$ Label	Offset (Hex)	Description
DSXPCTYP	+00	Call type (not a pointer). DSXFIRST = before printing DSXLAST = after printing
DSXPCNUM	+04	Call number (not a pointer). 1 is the first call for the above type.
DSXPPQCB	+08	Pointer to a copy of the Queue Control Block.
DSXPPJCB	+0C	Pointer to a copy of the Job Control Block.
DSXPPDCB	+10	Pointer to a copy of the SOLVE:EPS Dataset Control Block.
DSXPSTRL	+14	Pointer to the length (a fullword integer) of the return string. On entry, it is set to the maximum length that the return string can hold. Before returning, the exit must set it to the length of the data it has placed in the return string buffer.
DSXPSTRA	+18	Pointer to the return string buffer.
DSXPSTRO	+1C	Pointer to the return string options (a fullword integer).
DSXPPTM	+20	Pointer to time information structure.

Return

On return to SOLVE:EPS, an Assembler Dataset User Exit passes a return code in register 15. The following table describes the Assembler Dataset User Exit return codes:

Table A-13 Assembler Dataset User Exit Return Codes

OXDSCAL\$ Label	Hex Value	Description
DSXDATA	00	Indicates that SOLVE:EPS should insert the data from the return buffer into the output according to the return string options.
DSXPAGE	01	Indicates that SOLVE:EPS should perform the appropriate page break processing on the output. Note that the return string buffer and options are ignored for this return code.
DSXDONE	02	Indicates that SOLVE:EPS should not call the Dataset User Exit for this call type and data set again. Note that the return string buffer and options are ignored for this return code.

In addition, for return code DSXDATA the Dataset User Exit must set the length of the return buffer in the fullword integer pointed to by the DSXPSTRL parameter list entry.

It may also set return string processing options that are pointed to by DSXPSTRO; you can combine these return codes with an OR operation:

Table A-14 Return String Options

OXDSCAL\$ Label	Hex Value	Description
DSXTRANS	01	Translate.
DSXBLOCK	02	Insert as block letters.
DSXITALC	04	Insert as Italic letters (DSXBLOCK must also be set).
DSXLTERM	08	Line termination needed.

Time Structure

The time structure pointed to by the DSXPPTM parameter list entry consists of the fields shown in the following table. It is not set on each call, but on the first call of a particular type (DSXPCTYPE).

Table A-15 Time Information Structure

Offset (Hex)	Description
+00	Seconds after the minute (0-59)
+04	Minutes after the hour (0-59)
+08	Hours since midnight (0-23)
+0C	Day of the month (1-31)
+10	Month of the year (0-11)
+14	Years since 1900
+18	Days since Sunday (0-6)
+1C	Day of the year (0-365)
+20	Two-digit century
+24	Four-digit year

EPSUSEX Interface

When calling a Dataset User Exit that has been written in EPSUSEX (described in the [“EPSUSEX User Exit Language”](#) appendix), SOLVE:EPS makes some of its data structures available to the exit.

Basic Processing

To insert data into SOLVE:EPS output from a Dataset User Exit, set the fields of the returnstring object, and then set the return code to return data. The data returned will generally be based on the call type and call number.

To force a form feed, set the return code to return startpage.

To tell SOLVE:EPS not to call again, set the return code to return done.

Data Objects Available to the Dataset User Exit

The following table lists the objects that SOLVE:EPS makes available to an EPSUSEX Dataset User Exit:

Table A-16 Data Objects Available to Dataset User Exit

Object Name	Comments
queue	Selected fields from a copy of the queue data structure.
job	Selected fields from the data structure for a SOLVE:EPS job.
dataset	Selected fields from the data structure for a SOLVE:EPS data set.
call	Fields relating to a particular call of the Dataset User Exit. Details are given in the table following, “call Object Fields for the Dataset User Exit” on page A-22.
calltype	Constant fields for querying the calltype field. Details are given in the table following, “calltype Object Fields for the Dataset User Exit” on page A-24.
returnstring	Fields for the information that the exit returns, and that SOLVE:EPS is to insert into its output. Details are given in the table following, “returnstring Object Fields for the Dataset User Exit” on page A-24.
stringoptions	Constant fields for setting the returnstring options field. Details are given in the table following, “stringoptions Object Fields for the Dataset User Exit” on page A-26.
rc	A single field exit return code. Set its value with the return object.
return	A constant object containing fields for setting the rc object.
time	A collection of fields representing the date and time at which the Dataset User Exit is called. It is not set before each call, but rather once before the first call of a particular type.
boolean	A constant object holding true and false values for testing of boolean-type numeric fields.
jobdisposition	Constant fields used to examine jobdisposition values.
datatype	Constant fields used to examine job srcdatatype and job destdatatype values.
datasubtype	Constant fields used to examine job srcdatasubtype and job destdatasubtype values.

queue Object Fields for the Dataset User Exit

Selected fields from a copy of the queue data structure are made available to the Dataset User Exit. All of the fields are constant and cannot be modified.

Table A-17 Queue Object Fields for Dataset User Exit

Field Name	Data Type	Comments
outpname	string	The name of the outprocessor used by the queue.
description	string	A field that customers can assign to suit their own purposes, and that is not used by SOLVE:EPS directly.
nobanner	number	Determines whether a print request to a BSD lpd server suppresses the banner page control file line (the L line). Use the boolean object to interrogate this field.
localhost	string	A BSD-specific setting for local host name.
remotehost	string	A BSD-specific setting for remote host name.
remoteport	number	A BSD-specific setting for remote host lpr port.
remoteprinter	string	A BSD-specific setting for remote printer name.
filter	string	A BSD-specific setting for lpr filter.
sendallcopies	boolean	A sockets-specific (BSD and DSO Outprocessor) setting that indicates how to handle multiple-copy data sets. If true, the Outprocessor sends the complete data for each copy requested. If false, printer control is to be used to communicate the number of copies. For a BSD Outprocessor, control involves multiple print statements in the control file; for a DSO Outprocessor, control involves an escape sequence inserted by the data set user exit.

dataset Object Fields for The Dataset User Exit

Selected fields from the data structure for a SOLVE:EPS data set are made available to the Dataset User Exit. All of the fields are constant and cannot be modified.

Table A-18 Dataset Object Fields for Dataset User Exit

Field Name	Data Type	Comments
id	number	Dataset ID within the SOLVE:EPS job. This is the same as the job datasetnumber.
copies	number	Copy count
sourcespoolname	string	The spool data set name for the original file.

call Object Fields for the Dataset User Exit

This constant object contains information about the call to the Dataset User Exit. The following table describes the call fields.

Table A-19 Call Object Fields for Dataset User Exit

Field Name	Data Type	Comments
type	number	Call type. Use the calltype object to interrogate this field.
number	number	Current call number. The first call for a given call type is 1.

job Object Fields for the Dataset User Exit

Selected fields from the data structure for a SOLVE:EPS job are made available to the Dataset User Exit. All fields are constant and cannot be modified. The following table describes the job fields.

Table A-20 Job Object Fields for Dataset User Exit

Field Name	Data Type	Comments
id	number	SOLVE:EPS job number
datasetnumber	number	Data set number of the data set object within the SOLVE:EPS job. The first data set in a job is number 1.
datasetcount	number	The total number of data sets in this SOLVE:EPS job.
priority	number	SOLVE:EPS job priority (1-999).
disposition	number	Disposition of the SOLVE:EPS job after successful printing. Use the jobdisposition object to interrogate this field.
queue name	string	The name of the queue from which this job has been dispatched.
inpname	string	Name of the Inprocessor that created this job.
inptype	string	4-character type of the Inprocessor that created this job
srcdatatype	number	Source data type. Use the datatype object to interrogate this field.
srcdatasubtype	number	Source data subtype. Use the datasubtype object to interrogate this field.
destdatatype	number	Destination data type. Use the datatype object to interrogate this field.
destdatasubtype	number	Destination data subtype. Use the datasubtype object to interrogate this field.
jesjobnumber	number	JES job number for the original input. (Only meaningful for jobs coming from JES, i.e., an SSI or an NJE Inprocessor.)
trc	bit	Specifies that the data contains a Table Reference Character.
banner	bit	Specifies whether a banner page is to be produced by the remote (lpr) system.
indent	number	A BSD-specific setting for indentation. The value is placed on the I control file line.

Field Name	Data Type	Comments
width	number	Maximum line width. (For BSD outprocessors, this value is placed on the W control file line.)
class	string	Job class.
jobname	string	MVS job name of the originator of the print request. (For jobs coming from a VPT Inprocessor, this is the secondary LU name.)
notify	string	NOTIFY user ID
forms	string	FORMS
programmer	string	Programmer name
room	string	Room
department	string	Department
building	string	Building
remotemode	string	Remote node
remotedest	string	Remote destination
fcbl	string	FCB
ucs	string	UCS
writer	string	Writer
prmode	string	Processing mode
char1	string	Character arrangement table 1.
char2	string	Character arrangement table 2.
char3	string	Character arrangement table 3.
char4	string	Character arrangement table 4.
flash	string	Flash forms overlay
modificationid	string	Modification ID
pagedef	string	PSF PAGEDEF
formdef	string	PSF FORMDEF
originnode	string	Origin node
originuser	string	Origin user
accounting	string	Job accounting information
spoolbytes	number	The number of bytes this job occupies in SOLVE:EPS spool, i.e., in the temporary data set.
completions	number	A count of the number of times this job has been sent successfully.
resets	number	A count of the number of times this job has been RESET by the operator.
failures	number	A count of the number of times this job has been sent unsuccessfully.

calltype Object Fields for the Dataset User Exit

This constant object contains values for interrogating call object's type field. The following table describes the calltype fields.

Table A-21 Calltype Object Fields for Dataset User Exit

Field Name	Data Type	Comments
first	number	Indicates this call is before data set conversion
last	number	Indicates this call is after data set conversion

returnstring Object Fields for the Dataset User Exit

This object holds the information that the Dataset User Exit wants SOLVE:EPS to insert into the output. The following table describes the returnstring fields.

Table A-22 Returnstring Object Fields for Dataset User Exit

Field Name	Data Type	Comments
text	string	The data to insert into the SOLVE:EPS output. It can hold a maximum of 255 characters. The value of this field on entry to each call to the exit is an empty string.
options	number	Processing options for the above field. Use the stringoptions object to set this field (read "stringoptions Object Fields for the Dataset User Exit" on page A-26). The value of this field on entry to each call to the exit has no options set.

rc Object Fields for the Dataset User Exit

This is a single field Dataset User Exit return code. Use the return object to set its value

Table A-23 RC Object Fields for Dataset User Exit

Field Name	Data Type	Comments
value	number	The return code for the Dataset User Exit. Its value on entry to each call of the exit is return done.

boolean Object Fields for the Dataset User Exit

This is a constant object holding true and false values for testing of boolean-type numeric fields.

Table A-24 Boolean Object Fields for Dataset User Exit

Field Name	Data Type	Comments
true	number	A boolean true value (1).
false	number	A boolean false value (0).

return Object Fields for the Dataset User Exit

This is a constant object containing fields for setting the rc object in a Dataset User Exit.

Table A-25 Return Object Fields for Dataset User Exit

Field Name	Data Type	Comments
data	number	Indicates that SOLVE:EPS should insert returnstring text into the output according to the settings in returnstring options.
startpage	number	Indicates that SOLVE:EPS should perform the appropriate page break processing on the output. The returnstring object is ignored for this return code.
done	number	Indicates that SOLVE:EPS should not call the Dataset User Exit for this call type and data set again. The returnstring object is ignored for this return code.

time Object Fields for the Dataset User Exit

This is a constant object holding date and time information for use in a Dataset User Exit. These fields are set just before the call when call number == 1

Table A-26 Time Object Fields for Dataset User Exit

Field Name	Data Type	Comments
seconds	number	Seconds after the minute (0-59).
minutes	number	Minutes after the hour (0-59).
hour	number	Hours since midnight (0-23).
dayofmonth	number	Day of the month (1-31).
month	number	Month of the year (1-12).
century	number	Century (two digits).
year	number	Year (two digits only).
fullyear	number	Year (four digits).
dayofweek	number	Days since Sunday (0-6).
dayofyear	number	Day of the year (1-366).

jobdisposition Object Fields for the Dataset User Exit

This constant object contains values for interrogating a SOLVE:EPS job disposition. The term job disposition refers to what is to be done with a job after it prints successfully. The following table describes the jobdisposition fields:

Table A-27 Jobdisposition Object Fields for Dataset User Exit

Field Name	Data Type	Comments
purge	number	Indicates that SOLVE:EPS is to purge the job after successful printing.
hold	number	Indicates that SOLVE:EPS is to hold the job after successful printing.

datatype Object Fields for the Dataset User Exit

This is a constant object of values that can be used on the job srcdatatype and job destdatatype fields in a Dataset User Exit. The datasubtype object further refines the type of data. The following table describes the datatype fields:

Table A-28 Datatype Object Fields for Dataset User Exit

Field Name	Data Type	Comments
ascii	number	Indicates ASCII data.
ibmline	number	Indicates EBCDIC line data.
ibmlu	number	Indicates IBM LU-type data.

stringoptions Object Fields for the Dataset User Exit

A constant object for setting the options field of the returnstring object in a Dataset User Exit.

The following table describes the stringoptions fields.

Table A-29 Stringoptions Object Fields for Dataset User Exit

Field Name	Data Type	Comments
translate	number	Indicates that the returnstring text is to be translated.
blockletters	number	Indicates that the returnstring text is to be rendered as block letters in the output. The T07BLKFN module in the SOLVE:EPS load library concatenation contains the font information to use. Source code for this module is included in the SOLVE:EPS sample library for customer modification. (Implies terminateline.)
blocklettersitalic	number	Indicates that the returnstring text is to be rendered as italicized block letters in the output. Implies blockletters and terminateline.
terminateline	number	Indicates that the returnstring text is to have appropriate line termination processing performed.

datasubtype Object Fields for the Dataset User Exit

This is a constant object of values that can be used on the job srcdatasubtype and job destdatasubtype fields in a Dataset User Exit. The datatype object defines the major data type. The following table describes the datasubtype fields:

Table A-30 Datasubtype Object Fields for Dataset User Exit .

Field Name	Data Type	Comments
formatted	number	Indicates ASCII formatted data.
cif	number	Indicates ASCII CalTech Intermediate Form data.
dvi	number	Indicates ASCII DVI (TeX output) data.
graph	number	Indicates ASCII plot data.
leave	number	Indicates ASCII data leaving control characters.
ditroff	number	Indicates ASCII ditroff data.
postscript	number	Indicates ASCII Postscript data.
pr	number	Indicates ASCII data to be printed with the pr filter.
fortran	number	Indicates ASCII data with FORTRAN carriage control.
troff	number	Indicates ASCII troff data.
raster	number	Indicates ASCII data in Sun raster format.
asa	number	Indicates EBCDIC line data with ASA carriage control.
machine	number	Indicates EBCDIC line data with machine carriage control.
nocc	number	Indicates EBCDIC line data with no carriage control.
lu1	number	Indicates IBM LU type 1 data.
lu3	number	Indicates IBM LU type 3 data.

Calling the Dataset User Exit

The Dataset User Exit is called repeatedly before data set conversion and after conversion. On each call the exit can return information to insert into the output of the SOLVE:EPS job. It is the responsibility of the exit to inform SOLVE:EPS when it has completed inserting information.

Sample Dataset User Exits

Sample Dataset User Exits are distributed in the T07SAMP and T07PARM libraries.

Table A-31 Sample Dataset User Exits

Sample	Description
T07XX4AB	Assembler Dataset User Exit for a BSD outprocessor. It produces an MVS-like banner page at the beginning and at the end of a SOLVE:EPS job.
T07XX4EB	EPSUSEX Dataset User Exit for a BSD outprocessor. It produces an MVS-like banner page at the beginning and at the end of a SOLVE:EPS job.
T07XX4EJ	EPSUSEX Dataset User Exit for a JES or NJE outprocessor. It produces an MVS-like banner page at the beginning and at the end of a SOLVE:EPS job.
T07XX4HP	An EPSUSEX Dataset User Exit for a DSO Outprocessor. It uses the FORMS option to determine if landscape printing mode is desired. If it is, landscape printer controls are inserted in the output. It also handles the SENDALLCOPIES queue option, inserting copies control if SENDALLCOPIES is false.

Tips

The EPSUSEX code snippets provided in this section demonstrate some common tasks a Dataset User Exit might perform.

Setting Multiple returnstring Options

Much of the time the Dataset User Exit will need to set multiple return string options. This is accomplished by setting returnstring options to one of the stringoptions fields, and then adding other fields in. For example, to request translation and line termination, the following code would be used:

```
returnstring options := stringoptions translate.
```

```
returnstring options += stringoptions terminateline.
```

Note: Note that a particular stringoptions field should not be added more than once to the returnstring options.

Inserting Block Letters

Most banner pages feature block lettering of some sort. The Dataset User Exit can insert block letters in SOLVE:EPS output by setting the returnstring options field. The following example inserts the job name in block letters:

```
returnstring text := job jobname.
returnstring options := stringoptions blockletters.
returnstring options += stringoptions translate.
rc value := return data.
```

With block letters, it is not necessary to request line termination since it is implied.

If stringoptions blocklettersitalic had been used instead of using stringoptions blockletters, the block letters would have been italicized.

Source code for the block letter fonts is distributed with SOLVE:EPS. It is located in the T07BLKFN member of the T07SAMP library. If changes are made to it, it must be re-assembled and link-edited (RENT) into the SOLVE:EPS load library concatenation.

Determining the Beginning and the End of a SOLVE:EPS Job

Banner pages generally appear before the very first page of print data, and possibly after the very last page. The Dataset User Exit can determine at what point it is being called by looking at the call object and at the job's data set number field.

The following code returns without inserting data if the Dataset User Exit is not at the very beginning or the very end of the job:

```
if( call type == calltype first AND
    job datasetnumber != 1 OR
    call type == calltype last AND
    job datasetnumber != job datasetcount ) {
    rc value := return done.
    break.
}
```

Querying the Disposition Field

The following shows how to query the job disposition field:

```
if( job disposition != jobdisposition purge ) {
.
.
.
}
```

Querying Data Type and Subtype Fields

The following shows how to query a job's source and destination data types and subtypes:

```
if( job srcdatatype == datatype ibmlu AND
    job srcdatasubtype == datasubtype lu1 ) {
.
.
.
}
```

Querying the nobanner Field

The following shows how to query the queue nobanner field:

```
if( queue nobanner == boolean true ) {
.
.
.
}
```

Including Multiple Pieces of Information in returnstring Text

Some banner page lines contain more than a single piece of information. By using the EPSUSEX += string assignment operator, the returnstring object (see ["returnstring Object Fields for the Dataset User Exit" on page A-24](#)) can be built up from many pieces of information. This example follows some asterisks with the class, and then the word START or END:

```
returnstring text := '*****'.
returnstring text += job class.
if( call type == calltype first ) {
    returnstring text += ' START'.
}
else {
    returnstring text += ' END'.
}
returnstring options := stringoptions translate.
returnstring options += stringoptions terminateline.
rc value := return data.
```

Determining When to Insert What Data

Banner pages generally consist of more than one line of information. The Dataset User Exit decides what data to insert for any given call by examining the number field of the call object (read [“call Object Fields for the Dataset User Exit” on page A-22](#)). This example starts off with a form feed, followed by the job name in *italic block letters*, followed by some information about the job:

```
if( call number == 1 ) {
    rc value := return startpage.
}
elseif( call number == 2 ) {
    returnstring text := job jobname.
    returnstring options := stringoptions translate.
    returnstring options += stringoptions blocklettersitalic.
    rc value := return data.
}
elseif( call number == 3 ) {
    returnstring text := 'EPS JOB '.
    returnstring text += job id.
    returnstring options := stringoptions translate.
    returnstring options += stringoptions terminateline.
    rc value := return data.
}
else {
    rc value := return done.
}
```

Leading Zeros

When displaying a job number, it is customary to display leading zeros. EPSUSEX has no number formatting facilities, so adding leading zeros involves testing the magnitude of the number, and appending zeros to the returnstring object (read [“returnstring Object Fields for the Dataset User Exit” on page A-24](#)). This example inserts the SOLVE:EPS job number (with leading zeros) followed by the job class. The result is to be displayed in block letters.

```
returnstring text := 'J'.
if( job id < 10000 )           { returnstring text += '0'. }
if( job id < 1000 )           { returnstring text += '0'. }
if( job id < 100 )            { returnstring text += '0'. }
if( job id < 10 )             { returnstring text += '0'. }
returnstring text += job id.
returnstring text += ' '.
returnstring text += job class.
returnstring options := stringoptions translate.
returnstring options += stringoptions blockletters.
rc value := return data.
```

CONVERTSELECTOR Outprocessor User Exit

You can override the SOLVE:EPS converter selection by using a CONVERTSELECTOR user exit, which is assigned to an Outprocessor.

Following are two possible uses of this exit:

- To select the BINARY converter instead of LU1 conversion when the job has UCS=##NC as set by the LUASSOCIATION table.
- To select the Translate/Only converter based on the FORMS setting of the job.

See the description of the **SET OUTPROCESSOR** parameter “[CONVERTSELECTOR](#)” on page 1-56 for details.

An example of this user exit is shown in the sample CONVERTSELECTOR user exit, T07XX005. It implements LU1 passthru and selection of the Translate/Only converter. These converters are described in “System Architecture” (Chapter 2) in the ***SOLVE:EPS Installation and Administration Guide***.

The objects described in the following table are available to the CONVERTSELECTOR user exit:

Table A-32 Objects Available to CONVERTSELECTOR Outprocessor User Exit

Object	Description
QUEUE	Selected fields from the queue from which the job has been dispatched. See “ queue Object Fields for the Dataset User Exit ” on page A-21.
JOB	Selected fields from the data structure for a SOLVE:EPS job. See “ job Object Fields for the Dataset User Exit ” on page A-22.
BOOLEAN	A constant object holding true and false values for testing boolean-type numeric fields. See “ boolean Object Fields for the Dataset User Exit ” on page A-24.
DATATYPE	Constant fields used to examine job srcdatatype and job destdatatype values. See “ datatype Object Fields for the Dataset User Exit ” on page A-26.
DATASUBTYPE	Constant fields used to examine job srcdatasubtype and job destdatasubtype values. See “ datatype Object Fields for the Dataset User Exit ” on page A-26.
LANGUAGE	Constant fields used to examine job srclanguage and job destlanguage values. See “ language Object Fields ” on page A-54.
JOBDISP	Constant fields used to examine job disposition values. See “ jobdisposition Object Fields for the Dataset User Exit ” on page A-25.
CONVERTER	An object to hold the converter name. Initially it is set to the name of the converter that SOLVE:EPS has chosen. See “ converter Object Fields ” on page A-54.
RC	An object for setting the exit's return code and optional explanation. See “ rc Object Fields for CONVERTSELECTOR Outprocessor User Exit ” on page A-33.
RETURN	A constant object containing fields for setting the rc object. See “ return Object Fields for CONVERTSELECTOR Outprocessor User Exit ” on page A-33.

The converter object field shown in the following table describes the converter name for the CONVERTSELECTOR user exit:

Table A-33 Converter Name for CONVERTSELECTOR Outprocessor User Exit

Field Name	Data Type	Comments
name	string	This is the converter name. It must be a defined converter (see “DEFINE ALTERNATELOG” on page 1-6 for details). It is initially set to the converter selected by SOLVE:EPS, based on the job's source and destination data types and subtypes.

rc Object Fields for CONVERTSELECTOR Outprocessor User Exit

The object shown in this table contains fields for setting return values from the CONVERTSELECTOR Outprocessor User Exit.

Table A-34 RC Object Fields for CONVERTSELECTOR Outprocessor User Exit

Field Name	Data Type	Comments
value	number	Return code for the user exit. Use the return object to set it. It is initially set to okay.
explanation	string	Opportunity for you to explain the exit's actions. It can be a maximum of 128 characters. We recommend that you use this field if the exit rejects conversion.

return Object Fields for CONVERTSELECTOR Outprocessor User Exit

The constant object shown in the following table contains fields for setting the return code of the CONVERTSELECTOR user exit.

Table A-35 Return Object Fields for CONVERTSELECTOR Outprocessor User Exit

Field Name	Data Type	Comments
okay	number	Indicates that conversion may proceed with the converter name in the converter object.
notsupported	number	Indicates that this exit is rejecting conversion for this job. The job will be placed on error hold. If this return code is used, we recommend that the explanation field of the rc object be set.

Initialization User Exits

This section describes Initialization User Exits. There are two choices when you decide to add user exit processing to a converter:

- **SET** the INITEXIT parameter on an existing converter.
- **DEFINE** a new converter that has INITEXIT specified and select that converter from an Outprocessor CONVERTSELECTOR user exit.

LU1 Converter

The LU1 Converter Initialization User Exit may set a large number of conversion options, including default horizontal and vertical formats, initial and final form feed handling, and processing of individual code points. An example of the last option, is to specify that the LU1 converter is to treat an EBCDIC escape character (X'27') as a graphic character rather than its default as an erroneous SCS control code.

Most Useful LU1 Converter Initialization User Exit Options

The following table shows some of the most useful conversion options that can be set with the LU1 Converter Initialization User Exit:

Table A-36 LU1 Converter Initialization User Exit Options

Option	Comments
Horizontal Format	Left and right margins, maximum presentation position, and up to 32 horizontal tab positions.
Vertical Format	Top and bottom margins, maximum line position, and up to 32 vertical tab positions.
Form Feed	First and last form feed suppression or inclusion.
Graphic Error Code	The character printed for invalid SCS control codes.
Character Handling	Handling for each of the 256 EBCDIC single-byte codes.

An example of this is shown in the sample user exit T07XX006. A specific use of the LU1 Converter Initialization User Exit is described in the [“Overriding Character Handling for LU1” on page A-58](#).

The objects table [“Objects Available to the LU1 Converter Initialization User Exit” on page A-35](#) lists all of the objects available to the LU1 Converter Initialization User Exit. Following it are subtables describing some of the options in the objects table.

Objects Available to the LU1 Converter Initialization User Exit

The objects described in the following table are available to the LU1 Converter Initialization User Exit:

Table A-37 Objects Available to LU1 Converter Initialization User Exit

Object	Description
CONVERTER	Fields that provide information about the conversion environment. See “converter Object Fields” on page A-54 .
TRANSLATEDIRECTION	Fields that can determine the direction of translation for SOLVE:EPS converters. See “translatedirection Object Fields” on page A-56 .
DBCSUPPORT	Fields to determine the level of double-byte character set (DBCS) support for a SOLVE:EPS job. See “dbcssupport Object Fields” on page A-56 .
QUEUE	Selected fields from the queue from which the job has been dispatched. See “queue Object Fields for the Dataset User Exit” on page A-21 .
JOB	Selected fields from the data structure for a SOLVE:EPS job. See “job Object Fields for the Dataset User Exit” on page A-22 .
BOOLEAN	A constant object holding true and false values for testing of boolean-type numeric fields. See “boolean Object Fields for the Dataset User Exit” on page A-24 .
DATATYPE	Constant fields used to examine job srcdatatype and job destdatatype values. See “datatype Object Fields for the Dataset User Exit” on page A-26 .
DATASUBTYPE	Constant fields used to examine job srcdatasubtype and job destdatasubtype values. See “datatype Object Fields for the Dataset User Exit” on page A-26 .
LANGUAGE	Constant fields to examine job srclanguage and job destlanguage values. See “language Object Fields” on page A-54 .
JOBDISP	Constant fields to examine job disposition values. See “jobdisposition Object Fields for the Dataset User Exit” on page A-25 .
HORIZONTALFORMAT	Fields to set the default horizontal format for the job. See “horizontal Format Object Fields” on page A-38 .
VERTICALFORMAT	Fields to set the default vertical format for the job. See “vertical Format Object Fields” on page A-39 .
OPTIONS	Fields for various printing options. See “options Object Fields for the LU1 Converter Initialization User Exit” on page A-36 .
FORMFEED	Constant fields for form feed handling options. See “formfeed Object Fields” on page A-57 .
SCSCHARACTER	An indexed object of fields for specifying character handling options. See “scscharacter Object Fields” on page A-39 .
SCSTYPE	Constant fields for SCS character types. See “scstype Object Fields” on page A-40 .

Object	Description
EBCDIC	Constant fields for some EBCDIC characters. See “ebcdic Object Fields for the LU1 Converter Initialization User Exit” on page A-37 .
ASCII	Constant fields for some ASCII characters. See “ascii Object Fields for the LU1 Converter Initialization User Exit” on page A-38 .
RC	An object for setting the exit's return code and an optional explanation. See “rc Object Fields for Binary, JES-to-ASCII, LU1, and Translate/Only Initialization User Exits” on page A-55 .
RETURN	A constant object containing fields for setting the rc object. See “return Object Fields for Binary, JES-to-ASCII, LU1, and Translate/Only Initialization User Exits” on page A-55 .

options Object Fields for the LU1 Converter Initialization User Exit

The LU1 options object fields shown in the following table are constant object fields for LU1 conversion options:

Table A-38 Options Object Fields for LU1 Converter Initialization User Exit

Field Name	Data Type	Comments
graphicerrorcode	string	Character to use when an invalid SCS control is encountered. This character is translated before insertion into the output. The default value is dash.
cr	string	Character to use as a carriage return in terminating ASCII lines. This character is not translated before insertion into the output. The default value is an ASCII CR. By setting this field to a null string (X'00'), carriage return characters can be suppressed from the output.
lf	string	Character to use as a line feed in terminating ASCII lines. This character is not translated before insertion into the output. The default value is an ASCII LF.
ff	string	Character to use as a form feed. This character is not translated before insertion into the output. The default value is an ASCII FF.
firstformfeed	number	Handling option for an initial form feed. Use the formfeed object to interrogate and set this field. See “formfeed Object Fields” on page A-57 . The default value is asis.
lastformfeed	number	Handling option for a final form feed. Use the formfeed object to interrogate and set this field. See “formfeed Object Fields” on page A-57 . The default value is based on the queue's LASTFORMFEED setting.
compresscr	number	A flag that indicates whether carriage return characters are to be compressed in sequences of multiple line termination characters. For example, the sequence CRLF CRLF would become CRLF LF. Use the boolean object to interrogate and set this field. See “boolean Object Fields for the Dataset User Exit” on page A-24 . The default value is false.

Field Name	Data Type	Comments
nolineternbeforeff	number	A flag that indicates whether line termination characters before a form feed are to be suppressed. For example, the sequence CRLF CRFF would become CRFF. Use the boolean object to interrogate and set this field. See “boolean Object Fields for the Dataset User Exit” on page A-24 . The default value is false.

ebcdic Object Fields for the LU1 Converter Initialization User Exit

The ebcdic object fields shown in the following table contain fields for many EBCDIC control characters:

Table A-39 EBCDIC Object Fields for LU1 Converter Initialization User Exit

Field Name	Data Type	Comments	Field Name	Data Type	Comments
null	number	X'00'	soh	number	X'01'
stx	number	X'02'	etx	number	X'03'
sel	number	X'04'	ht	number	X'05'
rnl	number	X'06'	del	number	X'07'
ge	number	X'08'	sps	number	X'09
rpt	number	X'0A'	vt	number	X'0B'
ff	number	X'0C'	cr	number	X'0D'
so	number	X'0E'	si	number	X'0F'
dle	number	X'10'	dc1	number	X'11'
dc2	number	X'12'	dc3	number	X'13'
enp	number	X'14'	nl	number	X'15'
bs	number	X'16'	poc	number	X'17'
can	number	X'18'	em	number	X'19'
ubs	number	X'1A'	cu1	number	X'1B'
ifs	number	X'1C'	igs	number	X'1D'
irs	number	X'1E'	ius	number	X'1F'
ds	number	X'20'	sos	number	X'21'
fs	number	X'22'	wus	number	X'23'
inp	number	X'24'	lf	number	X'25'
etb	number	X'26'	esc	number	X'27'
sa	number	X'28'	sfe	number	X'29'
sw	number	X'2A'	csp	number	X'2B'
mfa	number	X'2C'	enq	number	X'2D'
ack	number	X'2E'	bel	number	X'2F'
hex30	number	X'30'	hex31	number	X'31'
syn	number	X'32'	ir	number	X'33'
pp	number	X'34'	trn	number	X'35'

Field Name	Data Type	Comments	Field Name	Data Type	Comments
ebs	number	X'36'	eot	number	X'37'
sbs	number	X'38'	it	number	X'39'
rff	number	X'3A'	cu3	number	X'3B'
dc4	number	X'3C'	nak	number	X'3E'
hex3e	number	X'3E'	sub	number	X'3E'
space	number	X'40'	rsp	number	X'41'
shy	number	X'CA'	esp	number	X'E1'

ascii Object Fields for the LU1 Converter Initialization User Exit

The ascii object fields shown in the following table contain fields for many ASCII control characters:

Table A-40 ASCII Object Fields for LU1 Converter Initialization User Exit

Field Name	Data Type	Comments
null	number	0x00
bel	number	0x07
bs	number	0x08
ht	number	0x09
lf	number	0x0A
vt	number	0x0B
ff	number	0x0C
cr	number	0x0D
exc	number	0x1B
space	number	0x20

horizontal Format Object Fields

The horizontal object fields shown in the following table are constant object fields for the horizontal format in the LU1 Converter Initialization User Exit:

Table A-41 Horizontal Format Object Fields for LU1 Converter Initialization User Exit

Field Name	Data Type	Comments
mpp	number	Maximum Presentation Position. The default value is the LINESIZE setting of the queue.
lm	number	Left Margin. The default value is 1.
rm	number	Right Margin. The default value is the LINESIZE setting of the queue.
tab <i>n</i>	number	A horizontal tab position. <i>n</i> ranges from 1 through 32. Each tab stop is initialized to an unset value (0).

vertical Format Object Fields

The vertical format object fields shown in the following table are constant object fields for the vertical format in the LU1 Converter Initialization User Exit:

Table A-42 Vertical Format Object Fields for LU1 Converter Initialization User Exit

Field Name	Data Type	Comments
mpl	number	Maximum Presentation Line. The default value is the PAGESIZE setting of the queue.
tm	number	Top Margin. The default value is 1.
bm	number	Bottom Margin. The default value is the PAGESIZE setting of the queue.
tab <i>n</i>	number	A vertical tab position. <i>n</i> ranges from 1 through 32. Each tab stop is initialized to an unset value (0).

scscharacter Object Fields

The scschar object fields shown in the following table contain fields for an SCS character.

Note: These are indexed objects (see the section [“Index Object References” on page B-5](#) for more information).

The index for these objects can take on values from 0 through 255. We recommend that you use the ebcidc object's fields as the index reference whenever possible:

Table A-43 Scscharacter Object Fields for LU1 Converter Initialization User Exit

Field Name	Data Type	Comments
type	number	The character type. Use the scstype object to interrogate and set this field. See “scstype Object Fields” on page A-40 for more details.
value	number	The character value to replace the original input character. This field is used only when the type field is either passthru or replace. The default value is zero, the null character.

scstype Object Fields

The scstype object fields shown in the following table contain fields for classifying SCS characters:

Table A-44 Scstype Object Fields for LU1 Converter Initialization User Exit

Field Name	Data Type	Comments
graphic	number	Indicates a graphic character, one that is translated and placed in the output.
control	number	Indicates an SCS control character, one that is acted on by the LU1 converter.
invalid	number	Indicates an invalid SCS control character, one that is replaced by the graphic error code before translation.
ignore	number	Indicates a character that is to be ignored by the converter. The LU1 converter has no characters of this type; only an Initialization User Exit sets it.
passthru	number	Indicates a character that is to be replaced by the scschar object's value field in the output, untranslated. The LU1 converter has no characters of this type; only an Initialization User Exit sets it.
replace	number	Indicates a character that is to be replaced by the scschar object's value field in the output, translated. The LU1 converter has no characters of this type; only an Initialization User Exit sets it.

Translate/Only Converter Initialization User Exit

The Translate/Only converter supports an Initialization User Exit whose main purpose is to override the indentation selected by the converter.

For example, one use of the Initialization User Exit might be to cause JES carriage control to be included in the data that is translated.

To assign an exit to the default binary converter, use this command:

SET CONVERTER TRANSLATEONLY INITEXIT *mem_name*

Syntax Description

mem_name Specifies the member in the PARMLIB DD library that contains the exit source.

An example of this is shown in the sample member T07XX007. This sample can also be used with the Binary converter exit.

For a description of the objects available to the Translate/Only Converter Initialization User Exit, see [“Objects Available to the Translate/Only and Binary Converter Initialization User Exit” on page A-42.](#)

Binary Converter Initialization User Exit

The binary converter now supports an Initialization User Exit. The objects available to the exit are the same as for the Translate/Only converter. Use the **DEFINE** or **SET CONVERTER** command to assign an Initialization User Exit.

To assign an exit to the default binary converter, use this command:

SET CONVERTER BINARY INITEXIT *mem_name*

Syntax Description

mem_name Specifies the member in the PARMLIB DD library that contains the exit source.

An example of this is shown in the sample member T07XX007. This sample can also be used with the Translate/Only converter exit.

For a description of the objects available to the Binary Converter Initialization User Exit, see [“Objects Available to the Translate/Only and Binary Converter Initialization User Exit” on page A-42](#).

Objects Available to the Translate/Only and Binary Converter Initialization User Exit

The objects shown in the following table are available to the Translate/Only and Binary Converter Initialization User Exits:

Table A-45 Objects Available to Translate/Only and Binary Converter Initialization User Exit

Object	Description
CONVERTER	Fields with conversion environment information. See “converter Object Fields” on page A-54 .
TRANSLATEDIRECTION	Fields to determine the direction of translation for SOLVE:EPS converters. See “translatedirection Object Fields” on page A-56 .
DBCSSUPPORT	Fields to determine the level of DBCS support for a SOLVE:EPS job. See “dbcssupport Object Fields” on page A-56 .
QUEUE	Selected fields from the queue from which the job has been dispatched. See “queue Object Fields for the Dataset User Exit” on page A-21 .
JOB	Selected fields from the data structure for a SOLVE:EPS job. See “job Object Fields for the Dataset User Exit” on page A-22 .
BOOLEAN	A constant object holding true and false values for testing boolean-type fields. See “boolean Object Fields for the Dataset User Exit” on page A-24 .
DATATYPE	Constant fields used to examine job srcdatatype and job destdatatype values. See “datatype Object Fields for the Dataset User Exit” on page A-26 .
DATASUBTYPE	Constant fields used to examine job srcdatasubtype and job destdatasubtype values. See “datatype Object Fields for the Dataset User Exit” on page A-26 .
LANGUAGE	Constant fields to examine job srclanguage and job destlanguage values. See “language Object Fields” on page A-54 .
JOBDISP	Constant fields to examine job disposition values. See “jobdisposition Object Fields for the Dataset User Exit” on page A-25 .
OPTIONS	Fields for conversion options. See “options Object Fields for the LU1 Converter Initialization User Exit” on page A-36 .
RC	An object for setting the exit’s return code and an optional explanation. See “rc Object Fields for Binary, JES-to-ASCII, LU1, and Translate/Only Initialization User Exits” on page A-55 .
RETURN	A constant object containing fields for setting the rc object. See “return Object Fields for Binary, JES-to-ASCII, LU1, and Translate/Only Initialization User Exits” on page A-55 .

JES-to-ASCII Converter

The JES-to-ASCII converter now supports an Initialization User Exit. The objects available to the exit can be used to manage the generation of ASCII carriage control. Use the **DEFINE** or **SET CONVERTER** command to assign an Initialization User Exit.

To assign an exit to the default JES-to-ASCII converter, use this command:

```
SET CONVERTER JES-TO-ASCS INITEXIT mem_name
```

Syntax Description

mem_name Specifies the member in the PARMLIB DD library that contains the exit source.

An example of this is shown in the sample member T07XX008.

The objects table [“Objects Available to the JES-to-ASCII Converter Initialization User Exit” on page A-44](#) lists all of the objects available to the LU1 Converter Initialization User Exit. Following it are subtables describing some of the options in the objects table.

Objects Available to the JES-to-ASCII Converter Initialization User Exit

The JES-to-ASCII object fields shown in the following table are constant object fields for JES-to-ASCII conversion options:

Table A-46 Objects Available to JES-to-ASCII Converter Initialization User Exit

Object	Description
CONVERTER	Fields that provide information about the conversion environment. See “converter Object Fields” on page A-54 .
TRANSLATEDIRECTION	Fields to determine the direction of translation for SOLVE:EPS converters. See “translatedirection Object Fields” on page A-56 .
DBCSUPPORT	Fields to determine the level of DBCS support for a SOLVE:EPS job. See “dbcssupport Object Fields” on page A-56 .
QUEUE	Selected fields from the queue from which the job has been dispatched. See “queue Object Fields for the Dataset User Exit” on page A-21 .
JOB	Selected fields from the data structure for a SOLVE:EPS job. See “job Object Fields for the Dataset User Exit” on page A-22 .
BOOLEAN	A constant object holding true and false values for testing of boolean-type numeric fields. See “boolean Object Fields for the Dataset User Exit” on page A-24 .
DATATYPE	Constant fields used to examine job srcdatatype and job destdatatype values. See “datatype Object Fields for the Dataset User Exit” on page A-26 .
DATASUBTYPE	Constant fields used to examine job srcdatasubtype and job destdatasubtype values. See “datatype Object Fields for the Dataset User Exit” on page A-26 .
LANGUAGE	Constant fields used to examine job srclanguage and job destlanguage values. See “language Object Fields” on page A-54 .
OPTIONS	Fields for various printing options. See “options Object Fields for the JES-to-ASCII Converter Initialization User Exit” on page A-45 .
FORMFEED	Constant fields for form feed handling options. See “formfeed Object Fields” on page A-57 .
EBCDIC	Constant fields for some EBCDIC characters. See “ebcdic Object Fields for the JES-to-ASCII Converter Initialization User Exit” on page A-46 .
ASCII	Constant fields for some ASCII characters. See “ascii Object Fields for the JES-to-ASCII Converter Initialization User Exit” on page A-46 .
RC	An object for setting the exit’s return code and an optional explanation. See “rc Object Fields for Binary, JES-to-ASCII, LU1, and Translate/Only Initialization User Exits” on page A-55 .
RETURN	A constant object containing fields for setting the rc object. See “return Object Fields for Binary, JES-to-ASCII, LU1, and Translate/Only Initialization User Exits” on page A-55 .

options Object Fields for the JES-to-ASCII Converter Initialization User Exit

The JES-to-ASCII options object fields shown in the following table are constant object fields for LU1 conversion options:

Table A-47 Options Object Fields for JES-to-ASCII Converter Initialization User Exit

Field Name	Data Type	Comments
cr	string	Character to use as a carriage return in terminating ASCII lines. This character is not translated before insertion into the output. The default value is an ASCII CR. By setting this field to a null string (X'00'), carriage return characters can be suppressed from the output.
lf	string	Character to use as a line feed in terminating ASCII lines. This character is not translated before insertion into the output. The default value is an ASCII LF.
ff	string	Character to use as a form feed. This character is not translated before insertion into the output. The default value is an ASCII FF.
firstformfeed	number	Handling option for an initial form feed. Use the formfeed object to interrogate and set this field. See “formfeed Object Fields” on page A-57 . The default value is asis.
lastformfeed	number	Handling option for a final form feed. Use the formfeed object to interrogate and set this field. See “formfeed Object Fields” on page A-57 . The default value is based on the queue's LASTFORMFEED setting.
compresscr	number	A flag that indicates whether carriage return characters are to be compressed in sequences of multiple line termination characters. For example, the sequence CRLFCRLF would become CRLFLF. Use the boolean object to interrogate and set this field. See “boolean Object Fields for the Dataset User Exit” on page A-24 . The default value is false.
nolinetermbeforeff	number	A flag that indicates whether line termination characters before a form feed are to be suppressed. For example, the sequence CRLFCRFF would become CRFF. Use the boolean object to interrogate and set this field. See “boolean Object Fields for the Dataset User Exit” on page A-24 . The default value is false.

ebcdic Object Fields for the JES-to-ASCII Converter Initialization User Exit

The ebcdic object fields shown in the following table contain fields for many JES-to-ASCII control characters:

Table A-48 EBCDIC Object Fields for JES-to-ASCII Converter Initialization User Exit

Field Name	Data Type	Comments
null	number	X'00'
ff	number	X'0C'
cr	number	X'0D'
nl	number	X'15'
lf	number	X'25'
space	number	X'40'

ascii Object Fields for the JES-to-ASCII Converter Initialization User Exit

The ASCII object fields shown in the following table contain fields for many JES-to-ASCII control characters:

Table A-49 ASCII Object Fields for JES-to-ASCII Converter Initialization User Exit

Field Name	Data Type	Comments
null	number	0x00
bel	number	0x07
bs	number	0x08
ht	number	0x09
lf	number	0x0A
vt	number	0x0B
ff	number	0x0C
cr	number	0x0D
exc	number	0x1B
space	number	0x20

Command Security User Exit

The Command security user exit authorizes use of SOLVE:EPS commands, particularly the SOLVE:EPS-Admin interface.

The Command Security User Exit module is loaded and its main entry point is called at SOLVE:EPS startup. The return value of that call points to a function pointer vector that makes command security calls. These vectors are described in the following sections.

As distributed, the T07XXCSY exit implements essentially the same SOLVE:EPS command security as the internal command security of SOLVE:EPS.

- Accepts all users; does not require valid user ID/password
- Accepts all commands from all consoles, including commands issued via SPSF
- Rejects the following commands from all alternate consoles:
 - **SHUTDOWN** (all flavors)
 - **STOP ALTERNATE CONSOLE**
 - **STOP ALTERNATECONSOLE SESSION**
 - **STOP** *anything* *
 - **CLEAR** *anything* *
 - **PURGE** *
 - **RELEASE** *
 - **RESET** *

Main Entry Point

The main entry point is entered with no arguments. It initializes the command security environment. It returns a pointer to this environment, the first word of which must point to the following list of function pointers:

VerifyUser	Verifies that a user has access to SOLVE:EPS for entering commands.
AuthorizeCommand	Authorizes use of a particular command for a verified user.
ReleaseUser	Indicates that a verified user is no longer going to enter SOLVE:EPS commands.
Terminate	Called when SOLVE:EPS is terminating and signals the end of the need for SOLVE:EPS command security.
IsEachSignonUnique	Answers whether this interface wants each user command access to SOLVE:EPS. In other words, signon WTD message, to be considered a unique user instance or not.

Note: Parameter lists to the above functions do not employ a VL bit to indicate their end.

VerifyUser

The VerifyUser call verifies that a user has access to SOLVE:EPS for entering commands. It is entered with the parameter list pointer in R1. The parameter list contains the following values:

- +0 Pointer to command security environment returned by the main entry point.
- +4 Pointer to signon info (see CMD\$CRTY macro).
- +8 Pointer to command origin info (see CMD\$CRTY macro).
- +C Pointer to a character string to hold a reject reason (64 bytes, null-terminated).
- +10 Pointer to a token. The meaning of the token is specific to the command user exit

This function sets the return code in R15, indicating success or failure (see CMD\$CRTY macro).

Note: A security token of -1 indicates that command authorization does not need to be called.

Note: If this function replaces an existing security token, the VerifyUser call is responsible for disposing of the replaced one.

AuthorizeCommand

The AuthorizeCommand call authorizes use of a particular command for a verified user. It is entered with the parameter list pointer in R1. The parameter list contains the following values

- +0 Pointer to command security environment returned by the main entry point.
- +4 Security token returned by VerifyUser.
- +8 Pointer to a command summary (see CMD\$CRTY macro).
- +C Pointer to a character string to hold a reject reason (64 bytes, null-terminated).

This function sets the return code in R15, indicating success or failure (see CMD\$CRTY macro).

ReleaseUser

The ReleaseUser call indicates that a verified user is no longer going to enter SOLVE:EPS commands. It is entered with the parameter list pointer in R1. The parameter list contains the following values:

- +0 Pointer to command security environment returned by the main entry point.
- +4 Security token returned by VerifyUser.

This function sets the return code in R15, indicating success or failure (see CMD\$CRTY macro).

Terminate

Terminate is called when SOLVE:EPS is terminating and signals the end of the need for SOLVE:EPS command security. It is entered with the parameter list pointer in R1. The parameter list contains the following values:

+0 Pointer to command security environment returned by the main entry point.

This call has no return.

IsEachSignonUnique

The IsEachSignonUnique call answers whether this interface wants each user command access to SOLVE:EPS (for example, signon WTD message) is to be considered a unique user instance or not. The response from this function determines what the VerifyUser function may see in the pointer to the token, VERIPSEC in CMD\$CRTY. An affirmative response guarantees that it will always be NULL (0); a negative response means that it may not be NULL. It is entered with the parameter list pointer in R1. The parameter list contains the following values:

+0 Pointer to command security environment returned by the main entry point.

This function sets the return code in R15, which must be one of the following:

- 0 NO, user command accesses after an initial access are to be considered instances of the initial access.
- 1 YES, each command access is to be considered an independent, unique access, regardless of whether the same user has other command access already or not.

Sample SAF Based Command Security Exit T07XXCS2

T07XXCS2 is a sample security exit distributed to illustrate how an installation may want to implement a SAF based command security exit. What follows describes how one would implement this exit as is.

As shipped, T07XXCS2 provides for user verification and resource authorization checking. User verification is the process of matching a User ID to a password. Resources authorization is the process of determining a verified user's authority to access a particular resource.

T07XXCS2 uses a RACROUTE REQUEST=VERIFY invocation to enable user verification. If RACROUTE returns a value of 0 in R15, the user is considered verified and the verification instance is now represented by a control block called an ACEE. The ACEE is subsequently used as input to resource authorization. If RACROUTE returns a non-zero value in R15, user verification is considered to have failed.

RACROUTE REQUEST=AUTH is used by T07XXCS2 to check a verified user's authorization to access a particular resource for a particular purpose. If RACROUTE returns a value not greater than 4 in R15, the resource access is allowed; otherwise, it is denied. A table in T07XXCS2 at label ALVLCMDV establishes a correlation between a SOLVE:EPS command and RACROUTE parameters ENTITY and ATTR. The table is searched sequentially. For instance, the SOLVE:EPS command **START INPROCESSOR** *myinp* maps to an entity name of INP.MYINP and an attribute of Update. RACROUTE would check if the user was authorized to resource INP.MYINP for update access. If a match in table ALVLCMDV is not found, T07XXCS2 bypasses the RACROUTE invocation and denies authorization.

Implementing Command Security Using RACF

T07XXCS2 may be used to engage RACF to authorize system and resource access. To use the provided sample exit T07XXCS2 essentially as is, proceed as follows:

1. Specify the RACF CLASS name you wish to use for SOLVE:EPS command security at label EPSCLASS. As shipped, this is ICS\$CEPS.
2. Specify the RACF PROFILE prefixes you wish to use for SOLVE:EPS command security starting at label PFXINP. These prefixes are prepended to an object name to derive an entity (or profile) name. If the prefix ends with a period (.), T07XXCS2 will derive a name in the form of *prefix.object_name*; if not, the prefix is used as is.
3. Specify the SOLVE:EPS command to RACF profile correlations that you wish to use at label ALVLCMDV.

Note: Consider this carefully. Once a methodology is established via this table and RACF profiles are created, it could take considerable effort to change.

Entries in this table are mapped by the ACCELEM DSECT:

ACCVERB	Specifies the SOLVE:EPS command verb being correlated. See macro \$CMDVERB for possible values. Examples of some command verbs are START, STOP, SET, and SHOW.
ACCTYPE	Specifies the SOLVE:EPS object type being correlated. See macro \$CMDOBJ for possible values. 0 is a catch all and causes a match when encountered. • Some SOLVE:EPS command verbs do not have an object type associated with them. The catch all must be used in these cases. These verbs are EXECUTE, HOLD, PROMPT, PURGE, REFRESH, RELEASE, RESET, and SHUTDOWN.
ACCLEVEL	Provides the verb/object correlation to a RACF resource authority level (READ, UPDATE, CONTROL, or ALTER). If 0 is specified, T07XXCS2 bypasses RACROUTE processing and simply authorizes the access.
ACCPFX	Provides the verb/object correlation to a RACF PROFILE prefix constant. The address of a PROFILE prefix is provided here. PROFILE prefixes should be 65 characters with blank padding.

4. Review return code checking after all RACROUTE calls. Some installations may want to respond differently to an RC4 (user profile not defined) from the RACROUTE REQUEST=VERIFY invocation, maybe establishing a default User ID for remote users that do not have a User ID on the local system. As well, the RACROUTE REQUEST=AUTH invocation may return a 4 when a profile cannot be found (if the system option PROTECTALL is not in force) and one may wish to fail these request rather than allow them.
5. Assemble and link T07XXCS2. Sample JCL is provided in member T07XXASM.
6. Modify the RACF Class Descriptor Table (ICHERCDE). Add the following entries to your RACF Class Descriptor Table. Refer to IBM publication *RACF Macros and Interfaces* for details about creating or updating a Class Descriptor Table.

```
ICS$CEPS ICHERCDE CLASS=ICS$CEPS,
        FIRST=ANY,
        GROUP=ICS$GEPS,
        ID=xxx,
        MAXLNTH=65,
        OTHER=ANY,
        POSIT=xxx,
        RACLIST=ALLOWED
```

```
ICS$GEPS ICHERCDE CLASS=ICS$GEPS,
        FIRST=ANY,
        ID=xxx,
        MAXLNTH=65,
        MEMBER=ICS$CEPS,
        OTHER=ANY,
        POSIT=xxx
```

Refer to the above mentioned IBM publication when determining your ID and POSIT values. You may wish to specify a DFTUACC value as well. The POSIT and ID values for the classes should be the same. Class ICS\$GEPS is a grouping class that allows a single profile to reference multiple members. This achieves the same result provided by RACF generic profiles without requiring resource names to match a pattern.

7. Modify the RACF Router Table (ICHRFRTB). Add the following entries to your RACF Router Table. Refer to IBM publication ***RACF Macros and Interfaces*** for details about creating or updating a Router Table.

```
ICS$CEPS ICHRFRTB CLASS=ICS$CEPS, ACTION=RACF
ICS$GEPS ICHRFRTB CLASS=ICS$GEPS, ACTION=RACF
```

8. Create the RACF profiles. Use the RACF RDEFINE and PERMIT commands to create the RACF profiles and their access lists. Refer to step 2) to ascertain the names of the profiles you need to create. Refer to IBM publications ***RACF Command Language Reference*** and ***RACF Security Administrator's Guide*** for information regarding the RDEFINE and PERMIT commands. If T07XXCS2 is used as is, the following profiles should be created as follows:

RDEFINE ICS\$CEPS SPOOL UACC(NONE)

This profile protects the SOLVE:EPS spool, including SOLVE:EPS jobs that reside on the spool. Examples of SOLVE:EPS commands that would be protected by this profile are **SET SPOOL**, **SET JOB**, **PURGE JOB**, etc.

RDEFINE ICS\$CEPS EPS UACC(NONE)

This profile protects SOLVE:EPS level commands, for instance; **PROMPT**, **SHUTDOWN**, **SET EPS**, etc.

RDEFINE ICS\$CEPS LOG UACC(NONE)

This profile protects the SOLVE:EPS log object. Examples of commands that would be protected by this profile are **SET LOG** and **SPIN LOG**.

As well, profiles should be defined to protect the remaining SOLVE:EPS object types. Generic profiles may be used if appropriate as well as grouping class profiles. Examples follow:

RDEFINE ICS\$CEPS INP.MYINP UACC(NONE)

This profile protects Inprocessor 'MYINP'.

RDEFINE ICS\$CEPS INP. UACC(NONE)**

This profile protects Inprocessors not protected by a more explicit profile.

RDEFINE ICS\$CEPS OUTP.OP* UACC(NONE)

This profile protects Outprocessors having a name beginning with 'OP'.

RDEFINE ICS\$CEPS ALTCONS.UACC(NONE)**

This profile protects Alternate Consoles not protected by a more explicit profile.

RDEFINE ICS\$CEPS EXEC.MYEXEC UACC(NONE)

This profile protects Exec member 'MYEXEC'.

RDEFINE ICS\$CEPS TABLE.TABLE2 UACC(NONE)

This profile protects table 'TABLE2'

RDEFINE ICS\$CEPS ** UACC(NONE)

This profile protects SOLVE:EPS objects of any type that are not protected by a more explicit profile.

RDEFINE ICS\$GEPS DEPT15_OBJECTS**ADDMEM(Queue.ABQ1,INP.CDE2,OUTP.CAT) UACC(NONE)**

This profile is a grouping profile that protects Queue 'ABQ1', Inprocessor 'CDE2', and Outprocessor 'CAT'.

RDEFINE ICS\$GEPS DEPT21_EXECS**ADDMEM(EXEC.A1,EXEC.B2,EXEC.C3) UACC(NONE)**

This profile is a grouping profile that protects Exec members 'A1', 'B2', and 'C3'.

PERMIT SPOOL CLASS(ICS\$CEPS) ID(JSMITH) ACCESS(ALTER)

This command grants user 'JSMITH' alter access to the SOLVE:EPS spool.

PERMIT DEPT15_OBJECTS CLASS(ICS\$CEPS) ID(DPT21) ACCESS(UPDATE)

This command grants users in RACF group 'DPT21' update access to Queue 'ABQ1', Inprocessor 'CDE2', and OUTProcessor 'CAT'.

9. Activate and RACLIST class ICS\$CEPS using the following command:

SETROPTS CLASSACT(ICS\$CEPS) RACLIST(ICS\$CEPS)

If the POSIT values (see step 6) for the member and grouping classes are the same, the above command will activate class ICS\$GEPS as well.

Response User Exit

This is sample code for a SOLVE:EPS Alternate Console response filter. It provides the opportunity to prevent the display of selected messages, e.g., trace and debug messages.

This filter blocks debug and trace level messages. The SOLVE:EPS e-Control functionality relies on this behavior, so it is strongly recommended that this filter be used, as is, with any Alternate Consoles that act as servers for e-Control.

This exit is entered with the parameter list pointer R1, which points to the following:

+0 Log message pointer (use \$LOGMSG to map).

Note: The VL bit is not set in the parameter list.

This call returns code R15, which points to the following:

0 Block the message from being displayed

^0 Display the message

Note: This same exit may be used with an e-Control Alternate Log object.

Shared Fields Tables

This section provides tables describing object fields that are common to the commands and other tables described in this document.

language Object Fields

The language object fields in the following table are constant objects that contain fields for interrogating a SOLVE:EPS job's srclanguage or job destlanguage:

Table A-50 Language Object Fields

Field Name	Data Type	Comments
unspecified	number	The default language; in other words, none has been specified.
none	number	No translation.
any	number	A wild card language type.

converter Object Fields

The converter object fields in the following table are converter Initialization User Exit constant object fields that provide information about the conversion environment:

Table A-51 Converter Object Fields

Field Name	Data Type	Comments
taskname	string	The name of the task in which the converter is running.
translating	number	Indicates whether translation is in effect or not. Use the boolean object to interrogate this field.
translatedirection	number	A value that indicates the direction of translation. Use the translatedirection object to interrogate this field. See “translatedirection Object Fields” on page A-56 .
dbcssupport	number	A value that indicates the level of double-byte character set (DBCS) translation support. Use the dbcssupport object to interrogate this field. See “dbcssupport Object Fields” on page A-56 .

rc Object Fields for Binary, JES-to-ASCII, LU1, and Translate/Only Initialization User Exits

The object described in the following table contains fields for setting return values from the Binary, JES-to-ASCII, LU1, or Translate/Only Initialization User Exits:

Table A-52 RC Object Fields for Binary, JES-to-ASCII, LU1, and Translate/Only Initialization User Exits

Field Name	Data Type	Comments
value	number	Return code from the user exit. Use the return object to set it. It is initially set to okay.
explanation	string	Opportunity for you to explain the exit's actions. It can be a maximum of 128 characters. We recommend that you use this field if the exit aborts conversion.

return Object Fields for Binary, JES-to-ASCII, LU1, and Translate/Only Initialization User Exits

The constant object described in the following table contains fields for setting the return code of the Binary, JES-to-ASCII, LU1, or Translate/Only Initialization User Exit:

Table A-53 Return Object Fields for Binary, JES-to-ASCII, LU1, and Translate/Only Initialization User Exits

Field Name	Data Type	Comments
okay	number	Indicates that conversion may proceed.
abort	number	Indicates that this exit is aborting conversion for this job. The job will be placed on error hold. We recommend that the explanation field of the rc object be set if this return code is used.

translatedirection Object Fields

The translatedirection object fields shown in the following table are constant object fields that can be used to determine the direction of translation for SOLVE:EPS converters:

Table A-54 Translatedirection Object Fields

Field Name	Data Type	Comments
ascii-to-ascii	number	Indicates ASCII-to-ASCII translation.
ascii-to-ebcdic	number	Indicates ASCII-to-EBCDIC translation.
ebcdic-to-ascii	number	Indicates EBCDIC-to-ASCII translation.
ebcdic-to-ebcdic	number	Indicates EBCDIC-to-EBCDIC translation.

dbcssupport Object Fields

The dbcssupport object fields shown in the following table are constant object fields that allow the determination of the level of double-byte character set (DBCS) support for a SOLVE:EPS job:

Table A-55 Dbcssupport Object Fields

Field Name	Data Type	Comments
none	number	Indicates no DBCS support.
sbc	number	Indicates single-byte support only.
euc	number	Indicates general DBCS support. (See Note, below.)
euc_japan	number	Indicates support for Japanese Extended UNIX Code (EUC) DBCS. This includes support for EUC-style single-byte katakana.
shift_jis	number	Indicates support for Japanese Shift JIS DBCS. This includes support for Shift JIS-style single-byte katakana.
euc_to_shift_jis	number	Indicates support for Japanese EUC-to-Shift JIS translation, including single-byte katakana. This level of DBCS support can only occur when the translatedirection is ASCII-to-ASCII. See “translatedirection Object Fields” on page A-56 .
shift-jis-to-euc	number	Indicates support for Japanese Shift JIS-to-EUC translation, including single-byte katakana. This level of DBCS support can only occur when the translatedirection is ASCII-to-ASCII. See “translatedirection Object Fields” on page A-56 .

Note: The term “general DBCS support” refers to ASCII DBCS encoding where the high-order bit of the first byte determines whether a code is single- or double-byte.

formfeed Object Fields

The formfeed object fields shown in the following table are constant object fields for form feed handling options:

Table A-56 Formfeed Object Fields

Field Name	Data Type	Comments
asis	number	Specifies no special form feed handling.
always	number	Specifies there is always to be a form feed. If the data does not contain one, a form feed is inserted.
never	number	Specifies there is never to be a form feed. If the data contains one, it is removed.

Overriding Character Handling for LU1

It is possible to override the default SOLVE:EPS character handling for LU1 conversion. For example, a character's classification can be changed from an SCS control code to a graphic character, it can be ignored, or it can be replaced by another character.

The general method for implementing customized character handling in LU1 conversion involves writing an EPSUSEX converter Initialization User Exit, and assigning it to the LU1 converter. This can be done in the following ways:

- Assign the user exit to the default SOLVE:EPS LU1 converter.
- Define a new LU1 converter, assign the user exit to it, and assign a CONVERTSELECTOR user exit to the Outprocessor to select the new converter.

In the following example, it is assumed that the user exit has been written, and has been placed in the PARMLIB DD data set.

Using the Default LU1 Converter

Using the default LU1 converter is the simplest way of getting customized LU1 character handling. Just assign the user exit member to the LU1 converter as shown in the following code:

SET CONVERTER LU1-TO-ASCS INITEXIT *mem_name*

Syntax Description

<i>mem_name</i>	Specifies the member name of the user exit in the PARMLIB DD data set. The best place to put this statement is the main SOLVE:EPS configuration file..
-----------------	--

Defining a New LU1 Converter

This method is more complex, but allows for more selectivity in the application of customized character handling. It involves the following steps:

- Define a new LU1 converter:

```
DEFINE CONVERTER LU1-SPECIAL INITMODULE T07CVI1B INITEXIT
```

```
mem_name
```

Syntax Description

mem_name Specifies the member name of the LU1 Converter Initialization User Exit. See the chapter [“SOLVE:EPS Operator Commands”](#) for more information.

- Write a CONVERTSELECTOR user exit.
- Assign the CONVERTSELECTOR user exit to the BSD Outprocessor with this command:

```
SET OUTPROCESSOR OBSD CONVERTSELECTOR mem_name2
```

Syntax Description

mem_name2 Specifies the name of the CONVERTSELECTOR user exit in the PARMLIB DD data set.

The best place to put these statements is the main SOLVE:EPS configuration file.

See the description for the CONVERTER object commands in the chapter [“SOLVE:EPS Operator Commands”](#).

Including Escape Sequences in LU1 Data

Some SOLVE:EPS customers have wanted to include escape sequences (such as HP PCL) into their LU1 data streams. If these sequences start with an EBCDIC escape character, the LU1 converter will process the character as an invalid SCS control code and substitute the graphic error code by default. This behavior will render this kind of escape sequence meaningless.

You can overcome this problem in the following ways:

- Have the LU1 converter treat the escape character as a regular graphic (printable) character.
- Have the LU1 converter insert an ASCII escape character into the output.

The following one-statement LU1 Converter Initialization User Exit instructs the converter to treat escape characters as printable data (the first method above):

scscharacter< ebcdic esc > type := scstype graphic.

This change in character classification causes EBCDIC escape characters to be translated according to the translate table in effect at conversion time.

The following two-statement user exit instructs the converter to insert an ASCII escape character into the output (the second method above):

scscharacter< ebcdic esc > type := scstype passthru.

scscharacter< ebcdic esc > value := ascii escape.

This change in character classification and value causes each EBCDIC escape character to be replaced by an ASCII escape character without translation.

Note: Both of these examples override EBCDIC escape character processing for all LU1 data streams. More coding would be necessary to be more selective, or to specify additional options.

Ignoring End-of-Message (EM) Characters

For this example, we assume that a customer has an LU1 application that is appending an LU3 End-of-Message (EM) character at the end of its RUs. To the LU1 converter these are invalid SCS control codes, and the graphic error code is substituted. Further, we assume that what the customer wants SOLVE:EPS to ignore the EM characters.

The following one-statement LU1 Converter Initialization User Exit tells the converter to ignore EM characters:

scscharacter< ebcdic em > type := scstype ignore.

As in the previous examples, LU1 converter behavior is overridden for all LU1 data streams. More coding would be necessary to be more selective, or to specify additional options.

Applying Character Handling Selectively

The above examples apply non-default LU1 conversion policies to all LU1 data streams they see. SOLVE:EPS provides two points at which to decide converter options:

- The Outprocessor CONVERTSELECTOR user exit

This exit selects a converter to use, and that converter would have an Initialization User Exit to set options.

- The converter Initialization User Exit

This exit would actually set the options.

To make decisions, there are a number of data structures available to converter Initialization User Exits, including the queue and the job. In the case of a job that has entered SOLVE:EPS via a VPT Inprocessor, the job originnode field holds the primary LU name, and the job originuser field holds the secondary LU1 name.

Normally, SOLVE:EPS jobs originating from a VPT Inprocessor do not have many job options set. However, customized job field values can be set in the LUASSOCIATION table. For example, to group a number of secondary LUs as ones requiring special handling for EBCDIC escape characters, the LUASSOCIATION table could set each applicable LU's UCS attribute to a certain value, such as #ESC. Then user exits, either the CONVERTSELECTOR or converter initialization, or both, could make decisions based on the job ucs field.

B

EPSUSEX User Exit Language

This appendix describes the User Exit Language (EPSUSEX). It includes these sections:

- [“Overview” on page B-2](#)
Provides a brief overview of the SOLVE:EPS User Exit Language.
- [“Data Types” on page B-2](#)
Describes the three major types of data that EPSUSEX can use.
- [“Data Categories” on page B-3](#)
Describes the two major categories of data that EPSU SEX uses.
- [“Index Object References” on page B-5](#)
Describes indexed objects, which let you access an array of objects.
- [“Statement Types” on page B-6](#)
Describes the statement blocks, or statement types, that EPSUSEX comprises.
- [“EPSUSEX Keywords” on page B-10](#)
Provides a list of EPSUSEX keywords.

Overview

SOLVE:EPS User Exit Language (EPSUSEX) is a simple, high level language in which some SOLVE:EPS user exits are implemented. Each exit makes available data structures suitable to its purposes, but all exits share a common syntax. This section describes that syntax.

When using EPSUSEX keywords and when referencing data structures, case does not matter. The values of the data referenced, however, is significant in most cases.

Data Types

The three major types of data that EPSUSEX can use are string, numeric and bit.

Strings

Strings are sequences of characters that are represented internally as C language strings. That is, they are terminated with a null character.

Numbers

Numbers are internal binary representations of integer numeric values. In object fields they can be either one, two, or four bytes in length. In literals, their range is from -8,388,608 to 8,388,607, inclusive (SOLVE:EPS does not normally work with negative or large values).

Bits

SOLVE:EPS represents some values in object fields by a single bit within a byte. They are accessed by referencing a bit-type object field. Comparison and assignment use EPSUSEX special constants on and off. Assignment can also make use of the toggle special constant.

Data Categories

EPSUSEX has two major categories of data: objects with fields and literals. Objects with fields allow access to SOLVE:EPS control structures, such as a queue or a job. Generally they are compared to literal values, and have literal values assigned to them.

Objects with fields

Objects with fields are accessed by a pair of names that are specific to particular user exits. For example, assume that a queue object is made available to a user exit, and that its description field is called description. To access that field, code the following:

queue description

Literals

Literals are values that are to be used as coded, without referring to any exit-provided data structures.

String Literals

String literals are enclosed in single-quotes. To include a single-quote in the string, precede it with a back slash.

This is an example of a string literal:

'AString'

Numeric Literals

Numeric literals are integer values, and can be expressed in either decimal or hexadecimal. A decimal expression is the familiar sequence of decimal digits. A hexadecimal expression begins with the characters 0x, followed by a sequence of hexadecimal digits.

This is an example of a numeric literal:

100

To represent the same number in hexadecimal, use the following:

0x64

Hexadecimal String literals

In situations where you want to use character strings that contain non-printable characters, hexadecimal string literals provide a way. The basic form of a hexadecimal string literal is to precede the opening apostrophe with the letter x, and to include only hexadecimal digits between the literal's apostrophes.

For example, the following code assigns EBCDIC <esc>E to the returnstring text field:

```
returnstring text := x'27C5'.
```

The append string operator, +=, is also supported.

Note: While it is possible to assign a string containing the null character, all zeros, you should be aware that any characters after the null character will be ignored. This is because EPSUSEX string types are C-style strings, which are terminated by a null character.

Bit Literals

Bit literals are represented by special keywords, and they can only occur on the right side of a comparison or assignment. The keyword ON refers to the bit represented by the corresponding object field in its on state. The keyword OFF refers to the bit represented by the corresponding object field in its off state. The keyword TOGGLE sets the bit represented by the corresponding object field to its opposite state. (TOGGLE can only be used in assignment statements.)

Index Object References

EPSUSEX supports indexed object references. Indexed objects let you access an array of objects.

Normally, when you access a piece of data through EPSUSEX, you specify an object/field name pair. For example, to access a job's jobname field, you would code the following:

job jobname

An indexed object requires an index reference after the object name. An index reference is either a number or a numeric object/field, enclosed in angle brackets. For example, to access the type of the EBCDIC escape character, code the following:

scschar < ebcdic escape > type

This example assumes there is an ebcdic object with a numeric escape field. To access the same data without using an object/field index, you would code the following:

scscharacter < 0x27 > type

Statement Types

EPSUSEX comprises statement blocks, which are statements enclosed in braces. The one exception is the initial statement block, which has no braces around it. There are four statement types:

- Conditionals to perform tests
- Assignments to change SOLVE:EPS data structure values
- The break statement to terminate user exit execution
- Comments

Two types of EPSUSEX statements do work in user exits. Conditionals test object field values, and assignments set object field values. Conditionals introduce embedded statement blocks, which can contain a mixture of assignment and nested conditional statements.

The meanings of both conditional tests and assignments depend on the individual exits and the data structures they make available.

Conditionals

Conditional statements are used to test object field values. They are introduced by the word **if** and are followed by a logical expression enclosed in parenthesis. A logical expression is made up of one or more tests; multiple tests are separated by logical operators **OR** and **AND**. Tests consist of a conditional comparator with values on either side of it. For example, to compare a job's priority to see if it is between 400 and 500, inclusive, you could use the following statement:

```
if( 400 <= job priority AND job priority <= 500 )
```

Conditionals are followed by statement blocks which are enclosed in braces.

if, elseif, and else Statements

Tests can be done with a single **if** statement, an **if** followed by an **else**, or an **if** followed by any number of **elseif** statements, possibly followed by an **else**. For example, to test a queue's description for one of three values, the code might look like this:

```
if( queue description == 'PCL' ) { ... }  
elseif( queue description == 'POSTSCRIPT' ) { ... }  
elseif( queue description == 'TEXT_ONLY' ) { ... }
```

Logical Expressions

Logical expressions consist of one or more comparisons using comparators separated by the logical operators OR and AND. The expected rules of precedence apply: AND takes precedence over OR. To override this precedence, logical subexpressions can be enclosed in parentheses. For example, in

```
if( queue description == 'PCL' AND  
   (job fcb == 'WIDE' OR job fcb == 'LAND') ) { ... }
```

the two tests of the job fcb field separated by the logical OR would be evaluated together. Without the inner parenthesis, the queue description and the first fcb test would be evaluated together, and then the results would be OR'ed with the second fcb test.

String Comparators

There are two sets of six string comparators. These are the case sensitive comparators:

- == Compare for equality.
- != Compare for inequality.
- >= Compare for left string value greater than or equal to right string value.
- <= Compare for left string value less than or equal to right string value.
- > Compare for left string value greater than right string value.
- < Compare for left string value less than right string value.

Case insensitive comparators are the above comparators enclosed in parentheses. For example, the following will match on Wide, WIDE, etc.:

```
job fcb (==) 'wide'
```

Numeric Comparators

Numeric comparators are the same as the case sensitive string comparators:

- == Compare for equality.
 - != Compare for inequality.
 - >= Compare for left numeric value greater than or equal to right numeric value.
 - <= Compare for left numeric value less than or equal to right numeric value.
 - > Compare for left numeric value greater than right numeric value.
 - < Compare for left numeric value less than right numeric value.
-

Bit Comparators

There are two bit value comparators:

<code>==</code>	equality
<code>!=</code>	inequality

The left side of the test must be an object field reference, and the right side must be one of the special constants on or off. For example, the following tests the trc bit field in the job object to see if it is set:

job trc == on

Assignments

Assignment statements change the values of SOLVE:EPS data structures available to a particular user exit. The left side of an assignment specifies the object to set, the right side specifies the value to assign, and the assignment operator in between specifies the type of assignment. Assignment statements always end in a period.

Note: Assignments are not verified for proper values, so exercise caution when changing SOLVE:EPS data structures.

String Assignment Operators

There are two string assignment operators:

<code>:=</code>	Set the object field on the left to the value on the right.
<code>+=</code>	Append the value on the right to the object field on the left.

It is not possible to overflow object fields, so values on the right side may be truncated to fit. For example, a job's class is a one character field, but the following would only set the class to A:

job class := 'ABCD'.

Number Assignment Operators

There are three numeric assignment operators:

<code>:=</code>	Set the object field on the left to the value on the right.
<code>+=</code>	Add the value on the right to the object field on the left, and store the result in the object field on the left.
<code>-=</code>	Subtract the value on the right from the object field on the left, and store the result in the object field on the left.

Bit Assignment Operators

Only straight assignments using the `:=` operator can be used against bit fields. The right side must be one of the special constants:

- `on` Turn on (set to 1) the object bit field specified on the left side.
 - `off` Turn off (set to 0) the object bit field specified on the left side.
 - `toggle` Change the state of the object bit field specified on the left side.
-

Assignment of Numbers to Strings

EPSUSEX now supports the assignment of number-type data to strings. It converts the number's value to displayable decimal digits, and assigns them to the string. Both string assignment operators, `:=` and `+=`, are supported.

For example, the following code sets the return string to the letter J followed by the SOLVE:EPS job number:

```
returnstring text := 'J'.
```

```
returnstring text += job id.
```

Note: EPSUSEX does not provide any formatting features other than the conversion of a number to displayable decimal. See tips for an example of adding leading zeros.

break Statement

The break statement can only appear at the end of a statement block, and it must be terminated by a period. break terminates the execution of the user exit at the end of the block in which it occurs, without executing any of the following statements.

Comments

A comment starts with two slash characters (`//`), and stops at the end of the line it appears on. As expected, comments are ignored during statement parsing.

EPSUSEX Keywords

The following is a list of EPSUSEX keywords:

and	Performs a logical AND operation on the results of two tests.
break	Stops EPSUSEX execution immediately.
else	Introduces default statement block.
elseif	Introduces an alternate conditional statement and its statement block.
if	Introduces a conditional statement and its statement block.
off	Tests or sets a bit field for an off value.
on	Tests or sets a bit field for an on value.
or	Performs a logical OR operation on the results of two tests.
toggle	Changes the state of a bit field to its opposite value.

Implemented Exits

The following user exits are implemented in EPSUSEX:

Table B-1 EPSUSEX User Exits

Exit	Object	Sample	Comments
Dispatch User Exit	queue	T01XX003	Read the description of the parameter “DISPATCHEXIT/NODISPATCHEXIT” on page 1-63 under the SET QUEUE command in the “SOLVE:EPS Operator Commands” chapter, and the section “Dispatch User Exit” on page A-6 in the “User Exits” appendix.
Dataset User Exit	queue	T07XX4EB	Read the description of the parameter “DATASETEXIT/NODATASETEXIT” on page 1-62 under the SET QUEUE command in the “SOLVE:EPS Operator Commands” chapter and the section “Dataset User Exit” on page A-15 in the “User Exits” appendix.
Convert Selector User Exit	Outprocessor	T07XX005	Read the description of the parameter “CONVERTSELECTOR” on page 1-56 under the SET OUTPROCESSOR command in the “SOLVE:EPS Operator Commands” chapter, and the section “CONVERTSELECTOR Outprocessor User Exit” on page A-32 in the “User Exits” appendix.

Exit	Object	Sample	Comments
LU1 Converter Initialization User Exit	Converter	T07XX006	Read the description of the command “SET CONVERTER” on page 1-27 in the “SOLVE:EPS Operator Commands” chapter and the section “Initialization User Exits” on page A-34 in the “User Exits” appendix.
Translate/Only and Binary convertor Initialization User Exits	Converter	T01XX007	Read the description of the command “SET CONVERTER” on page 1-27 in the “SOLVE:EPS Operator Commands” chapter and the section “Translate/Only Converter Initialization User Exit” on page A-40 in the “User Exits” appendix.
JES Converter Initialization User Exit	Converter	T01XX008	Read the description of the command “SET CONVERTER” on page 1-27 in the “SOLVE:EPS Operator Commands” chapter and the section “JES-to-ASCII Converter” on page A-43 in the “User Exits” appendix.

C

Using SOLVE:EPS-Administration

SOLVE:EPS-Administration (SOLVE:EPS-Admin) is an ISPF based application that allows one to select options and to specify parameters, commands, and program function (PF) keys to simplify requests for common functions.

This appendix includes the following sections:

- [“Overview” on page C-3](#)
Describes implementation of SOLVE:EPS-Admin.
- [“Commands” on page C-4](#)
Describes the SOLVE:EPS-Admin commands.
- [“Parameters” on page C-6](#)
Describes the SOLVE:EPS-Admin parameters, which let you view and manipulate SOLVE:EPS parameters.
- [“Queues and Outprocessors” on page C-8](#)
Describes the SOLVE:EPS-Admin queues and Outprocessors, which let you manipulate SOLVE:EPS queues and Outprocessors.
- [“Jobs” on page C-10](#)
Describes the SOLVE:EPS-Admin Jobs, which let you view and manipulate SOLVE:EPS jobs.
- [“Inprocessors” on page C-11](#)
Describes the SOLVE:EPS-Admin Inprocessors, which let you view and manipulate SOLVE:EPS Inprocessors.
- [“Spool” on page C-11](#)
Describes the SOLVE:EPS-Admin spools, which let you view and manipulate SOLVE:EPS spool characteristics.
- [“Logs” on page C-11](#)
Describes the SOLVE:EPS-Admin Logs, which let you view and manipulate SOLVE:EPS log characteristics.
- [“SOLVE:EPS” on page C-12](#)
Describes the SOLVE:EPS-Admin SOLVE:EPS utility, which lets you view and manipulate SOLVE:EPS characteristics.

-
- [“SOLVE:EPS Command Shell” on page C-12](#)

Describes the SOLVE:EPS command shell.

- [“Frequently Asked Questions” on page C-13](#)

Describes some of the most frequently asked questions and most frequently received error messages.

Overview

SOLVE:EPS-Admin is a user friendly full screen interface that lets you remotely administer an SOLVE:EPS address space using client/server technology. It replaces the command oriented line mode interface that required you to log on to the system on which SOLVE:EPS is running.

SOLVE:EPS-Admin is implemented as a network client; the server takes the form of an SOLVE:EPS object called an Alternate Console. Therefore, SOLVE:EPS-Admin can access SOLVE:EPS systems running on any host, provided an appropriately configured alternate console has been defined and started, and the host is accessible across the network.

As one negotiates SOLVE:EPS-Admin, data is requested from the target SOLVE:EPS system. When data is gathered for a particular panel display, it does not refresh when the Enter key is pressed. A **REFRESH** command is provided to refresh the data displayed on a panel. As shipped, function key **F5** invokes the **REFRESH** command where appropriate.

When one engages a dialog in a downward direction (in other words, by entering action characters or SOLVE:EPS-Admin commands, or selecting options), subsequent panel displays will contain freshly gathered data. As one returns back up a dialog path (in other words., by entering **END**, **EXIT**, or **CANCEL** commands, or pressing the Enter key in some instances), panels that are redisplayed contain stale data. Use the **REFRESH** command to replace stale data.

Commands

With a few exceptions, SOLVE:EPS-Admin commands can be entered from any SOLVE:EPS-Admin command line. Most of the commands are assigned to function keys as well (if the function key definitions have not been modified).

EXECUTE

The **EXECUTE** command lets you specify a member that contains a collection of SOLVE:EPS commands to be executed. The member resides on the system upon which SOLVE:EPS is executing. The **EXECUTE** command takes one optional parameter—the name of the member to be executed. For example:

Command ==> **execute exec06**

If the member name parameter is not specified, you will be prompted for the member name.

The **EXECUTE** command, as shipped, is not assigned to a function key.

NEWQUEUE

The **NEWQUEUE** command lets you create an SOLVE:EPS queue. It takes no parameters. As shipped, **NEWQUEUE** is available via function key from the Queues and Outprocessors panel. It is also invoked as a result of the **NEWQUEUE** action character from the same panel.

OPTIONS

The **OPTIONS** command lets you work with SOLVE:EPS-Admin parameters. The **OPTIONS** command takes no parameters. As shipped, **OPTIONS** is not assigned to a function key. **OPTIONS** is also invoked by selecting **Option 0** from the SOLVE:EPS-Admin Main Menu.

RELOAD

The **RELOAD** command lets you refresh various SOLVE:EPS tables. It takes no parameters. As shipped, **RELOAD** is available via function key from the SOLVE:EPS panel.

REFRESH

The **REFRESH** command causes data on a panel display to be re-fetched from the target SOLVE:EPS system. The **REFRESH** command takes no parameters. As shipped, **REFRESH** is available via function key where appropriate.

SHUTDOWN

The **SHUTDOWN** command lets you terminate a running SOLVE:EPS system. The **SHUTDOWN** command takes no parameters. As shipped, **SHUTDOWN** is available via function key from the SOLVE:EPS panel.

SPIN

The **SPIN** command lets you spin the SOLVE:EPS log. The **SPIN** command takes no parameters. As shipped, **SPIN** is available via function key from the Log panel.

Parameters

SOLVE:EPS-Admin parameters let you view and manipulate an active SOLVE:EPS-Admin session. Information on specifying the following field values is presented in sequence, or may be selected by number:

TCP Address Space Name

TCP Address Space Name specifies the jobname of a local address space that implements a TCP/IP stack. SOLVE:EPS-Admin uses the specified TCP address space when establishing a network connection to an SOLVE:EPS alternate console.

Remote Machine Name

Remote Machine Name specifies a remote TCP host on which an SOLVE:EPS alternate console has established a listening port. SOLVE:EPS-Admin uses the specified remote machine name when establishing a network connection to an SOLVE:EPS alternate console.

Remote Port

Remote Port specifies a remote TCP port on which an SOLVE:EPS alternate console is listening. SOLVE:EPS-Admin uses the specified remote port when establishing a network connection to an SOLVE:EPS alternate console.

Remote User ID

Remote User ID specifies a User ID that, in conjunction with Remote Password, can be used by the target SOLVE:EPS system to determine whether or not you are authorized to access the host on which it is running, and whether or not you may perform certain actions on certain SOLVE:EPS objects.

Remote Password

Remote Password specifies a password that, in conjunction with Remote User ID, can be used by the target SOLVE:EPS system to determine whether or not you are authorized to access the host on which it is running, and whether or not you may perform certain actions on certain SOLVE:EPS objects.

Inactivity Timeout

Inactivity Timeout specifies how long SOLVE:EPS-Admin will wait to complete a network connection, read, or write. Choose a value appropriate to your network's behavior.

Debug

Debug specifies the SOLVE:EPS-Admin debug level:

None	Perform no debugging.
Commands	Commands to and responses from SOLVE:EPS are printed to the screen
Low-level	Perform low-level debugging, in addition to command debugging.
Communications	Perform network interface debugging, in addition to low-level and command debugging (which can produce great amounts of output)

Native Interface Modeling

Use **Native Interface Modeling** when defining new objects specifies whether or not SOLVE:EPS models are to be used when defining new objects, such as queues. If this option is taken, upon entering the new object dialog, you will be prompted for the name of a model object. If provided, SOLVE:EPS-Admin will use it to override object defaults upon display of the define object panel. If the model does not exist, an error will result. If not provided, SOLVE:EPS-Admin will use the default model (if one exists) to over-ride object defaults.

Your installation may wish to control use of this option, and may do so in one of two ways. It may elect to not allow specification of this option, or, it may elect to not allow non-specification of this option. Attempts to change the option to a non-sanctioned value will result in errors.

Queues and Outprocessors

Queues and Outprocessors let you view and manipulate SOLVE:EPS queues and Outprocessors. You can change maximum sessions directly from the Queues and Outprocessors panel by typing over the existing value.

Queue Filter

The **Queue Filter** controls which queues and Outprocessors are presented on the Queues and Outprocessors panel. The default filter value is the wildcard character '*', which, when specified by itself, causes all defined queues and Outprocessors to be displayed. The wildcard character may also be specified as the last character of a string, in which case only queues that begin with the wildcarded string, and their associated Outprocessors, will be displayed. If a wildcard character is not employed, only the specific queue matching the filter string (if any), and its associated Outprocessor, will be displayed. The Queue Filter value may be changed at will, and is utilized in a case-sensitive fashion. Imbedded wildcard characters are employed literally.

Dummy Outprocessor

In the event that a queue references an invalid Outprocessor, that is, either a non-existent or model Outprocessor, a **Dummy Outprocessor** entry will be employed by the Queues and Outprocessors panel. The Dummy Outprocessor entry can be recognized by its name, *** INVALID OUTP ***, and may not be manipulated. The Dummy Outprocessor entry exists solely to maintain the Outprocessor/Queue relationship required by the Queues and Outprocessors panel.

Zoom Outprocessor

Zoom Outprocessor lets you view an Outprocessor's settings, change an Outprocessor's settings, and view an Outprocessor's counters.

- **View/Change Outprocessor Settings** — View and change an Outprocessor's settings.
 - **View/Change BSD Outprocessor Settings** – View and change settings for a BSD queue.
 - **View/Change DSO Outprocessor Settings** – View and change settings for a DSO Outprocessor.
 - **View/Change JES Outprocessor Settings** – View and change settings for a JES Outprocessor.
 - **View/Change NJE Outprocessor Settings** – View and change settings for an NJE Outprocessor.
- **View Outprocessor Counters** — View an Outprocessor's counters.

Zoom Queue

Zoom Queue lets you view a queue's settings, change a queue's settings, and view a queue's counters.

- **View/Change Queue Settings** — View and change a queue's settings.
 - **View/Change BSD Queue Settings** — View and change a BSD queue's settings.
 - **View/Change DSO Queue Settings** — View and change a DSO queue's settings.
 - **View/Change JES Queue Settings** — View and change a JES queue's settings.
 - **View/Change NJE Queue Settings** — View and change an NJE queue's settings.
 - **View Queue Counters** — View a queue's counters.
-

New Queue

New Queue lets you create a queue.

- **Specify Model Queue** lets you specify a model queue to be used during the creation of a new queue. If a model queue is not specified, the default model queue, if one is designated, will be used.
 - **New BSD Queue** — Create a BSD queue.
 - **New JES Queue** — Create a JES queue.
 - **New NJE Queue** — Create an NJE queue.
-

Route Queue

Route Queue lets you route jobs from one queue to another queue. Once a queue is routed, all existing jobs on the source queue are placed on the target queue. Newly created jobs destined for the source queue will also be placed on the target queue. Jobs that are active at the time the route is established will be placed on the target queue if they complete in error or with a disposition of hold.

To allow multi-hop routing, specify the force option. This allows a queue to participate in a route involving more than 2 queues. For instance, if Queue Foobar is routed to Queue Gizmo and one wishes to route Queue Gizmo to Queue AlleyOOp, the force option must be specified.

To "unroute" a queue, route the queue to itself.

Route processing prevents the occurrence of a routing loop, so, if Queue Foobar is routed to Queue Gizmo, an attempt to then route Queue Gizmo to Queue Foobar will result in an error.

Jobs

Jobs lets you view and manipulate SOLVE:EPS Jobs. The Jobs are presented in dispatch order; sorted by Priority, and Creation Time. Priority and queue may be changed directly from the Jobs panel by simply over-typing their present values.

Job Filters

Job Filters control which Jobs are presented on the Jobs panel. The default filter values are a wildcard character '*', which, when specified by itself, causes a match on all values for the field that the filter pertains to. The wildcard character may also be specified as the last character of a string, in which case only field values that begin with the wildcarded string will be matched. If a wildcard character is not employed, only the specific field value that equals the filter string will match. The Job Filter values may be changed at will, and are utilized in a case-sensitive fashion. Imbedded wildcard characters are employed literally.

Reset Job

Reset Job terminates an active job. The disposition of the terminated job is stipulated as either Hold, Ready, or Purge.

Zoom Job

Zoom Job lets you view a Job's settings, change a job's settings, view a Job's attributes, and view a job's counters.

- **View/Change Job Settings** — View and change a Job's settings.
- **View Job Attributes** — View a Job's attributes.
- **View Job Counters** — View a Job's counters.

Inprocessors

Inprocessors let you view and manipulate SOLVE:EPS Inprocessors. Maximum sessions may be changed directly from the Inprocessors panel by simply over-typing the present value.

Zoom Inprocessor

Zoom Inprocessor lets you view an Inprocessor's settings, change an Inprocessor's settings, and view an Inprocessor's counters.

- **View/Change Inprocessor Settings** — View and change settings for an Inprocessor.
 - **View/Change BSD Inprocessor Settings** — View and change settings for a BSD Inprocessor.
 - **View/Change NJE Inprocessor Settings** — View and change settings for an NJE Inprocessor.
 - **View/Change SSI Inprocessor Settings** — View and change settings for an SSI Inprocessor.
 - **View/Change VPRT Inprocessor Settings** — View and change settings for a VPRT Inprocessor.
- **View Inprocessor Counters** — View counters for an Inprocessor.

Spool

Spool lets you view and manipulate SOLVE:EPS Spool characteristics.

- **View/Change Spool Settings** — View and change SOLVE:EPS spool settings.
- **View Counters: Attempted Job Creation** — View SOLVE:EPS spool counters related to job creations (Inprocessor activity).
- **View Counters: Job Dispatch** — View SOLVE:EPS spool counters related to job dispatches (Outprocessor activity).

Logs

Log lets you view and manipulate SOLVE:EPS log characteristics.

- **View/Change EPS Settings** — View and change SOLVE:EPS log settings.

SOLVE:EPS

SOLVE:EPS lets you view and manipulate SOLVE:EPS characteristics.

- **View/Change EPS Settings** — View and change SOLVE:EPS settings.
- **View EPS Counters** — View SOLVE:EPS counters.
- **Reload Table** — Refresh and reload an SOLVE:EPS table.
- **Shutdown EPS** — Shut down SOLVE:EPS.

SOLVE:EPS Command Shell

EPS Command Shell lets you enter SOLVE:EPS commands. Responses display in line mode.

Frequently Asked Questions

This section describes some of the most Frequently Asked Questions (FAQs) and most frequently received error messages. There is also a **FAQ** option on the SOLVE:EPS-Admin Main Menu. This **FAQ** in SOLVE:EPS-Admin is dynamic, but this section of the documentation describes only the most frequently asked of questions (at the time of publication). So if you have a question that is not described here, try the **FAQ** option in the SOLVE:EPS-Admin Main Menu.

Error Messages

Communication failure - T070J OMXCT002 *ERR* COMM INITAPI request failed for Local Host code 10112

Explanation: This usually means that you have specified either an invalid TCP address space name or that the one specified is not up and running. The TCP address space name is settable via **Option 0** in the SOLVE:EPS-Admin Main Menu. If you know the value to be correct, ensure that the TCP address space is up and functional.

Communication failure - T070J OMXCT002 *ERR* COMM GETHOST request failed for Local Host code 00049

Explanation: This typically means that you have specified an invalid remote machine name. The **Remote machine name** is an option in the SOLVE:EPS-Admin Main Menu, settable via **Option 0**, and specifies a TCP host on which an SOLVE:EPS alternate console has established a listening port. Ensure that the value you have specified for remote machine name is correct.

Communication failure - T070J OMXCT005 *WRN* TIMEOUT (Request Timed-Out)

Explanation: If SOLVE:EPS-Admin is attempting to establish a connection to an SOLVE:EPS alternate console, this could indicate that the remote host (specified by SOLVE:EPS-Admin parameter **Remote machine name**) is down. Ensure that the remote host is up and functional.

If SOLVE:EPS-Admin has already established a connection with an SOLVE:EPS alternate console, this indicates that a network read or write did not complete in a timely fashion. This can be the result of temporary network congestion or an Inactivity timeout value that is too low.

Communication failure - T070J OMXCT011 *ERR* COMM Connection Refused

Explanation: This means that the target SOLVE:EPS system you are attempting to administer is either not up, or does not have an alternate console listening on the **Remote port** that you have specified. Ensure that the target SOLVE:EPS system is up, and has an alternate console of type TCP varied on and listening on the remote port that you have specified via **Option 0** of the SOLVE:EPS-Admin Main Menu.

Other Communication Failures

If you receive a communication failure message other than those described above, communication errors that have not already been addressed in this **FAQ** all have a code associated with them. This code is explained in one of the following IBM publications:

- *TCP/IP V3R1 for MVS: Application Programming Interface*
- *TCP/IP V3R2 for MVS: API Reference Appendix D*. Return Codes documents the error conditions.

Usage Notes

- Use the **REFRESH** command to cause SOLVE:EPS-Admin to refetch data from SOLVE:EPS.
When you press Enter to redisplay a given panel, or, when you type a command or press a function key that performs an END, EXIT, or CANCEL (which pops you back to the previously displayed panel), the data that displays may not seem up to date. This is because pressing the Enter key does not cause SOLVE:EPS-Admin to refetch data from the target SOLVE:EPS system. Neither does popping back a screen via END, EXIT, or CANCEL. As shipped, function key F5 has the **REFRESH** command assigned to it, where appropriate.
When proceeding forward through SOLVE:EPS-Admin, by entering commands or action characters, or by selecting options, data is freshly retrieved from the target SOLVE:EPS system prior to formatting the panel display. **REFRESH** is only required when you want to stay on a particular panel, or proceed back up through previously displayed panels, and want to freshen up stale data.
- The help panels and tutorial are designed to address topics specific to SOLVE:EPS-Admin. For information about topics not specific to SOLVE:EPS-Admin, such as SOLVE:EPS system concepts, read other chapters of this document.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Index

Symbols

/etc/printcap [3 - 6](#)

A

abbreviations [1 - 2](#)

ACCOUNT parameter, SET INPROCESSOR
command [1 - 36](#)

ACN, alias for ALTERNATECONSOLE command [1 - 6](#), [1 - 18](#), [1 - 23](#), [1 - 86](#), [1 - 88](#)

aliases

ACN, alias for ALTERNATECONSOLE [1 - 6](#), [1 - 18](#), [1 - 23](#), [1 - 86](#), [1 - 88](#)

ALTCONS, alias for ALTERNATECONSOLE
command [1 - 6](#), [1 - 18](#), [1 - 23](#), [1 - 86](#), [1 - 88](#)

AVGREC, alias for AVERAGERECORDS
parameter of SET SPOOL command [1 - 82](#)

CFILE, alias for CONTROLFILE parameter of
SET QUEUE command [1 - 70](#)

CREATE INPROCESSOR, alias for DEFINE
INPROCESSOR command [1 - 7](#)

CREATE OUTPROCESSOR, alias for DEFINE
OUTPROCESSOR command [1 - 8](#)

CREATE QUEUE, alias for DEFINE QUEUE
command [1 - 9](#)

CTLFILE, alias for CONTROLFILE parameter of
SET QUEUE command [1 - 70](#)

CTLSTR, alias for CONTROLSTRING
parameter of REFRESH command [1 - 14](#)

CTLSTRPRE, alias for
CONTROLSTRINGPREFIX parameter of
SET QUEUE command [1 - 71](#), [1 - 78](#)

CTLSTRSUF, alias for
CONTROLSTRINGSUFFIX parameter of
SET QUEUE command [1 - 72](#), [1 - 78](#)

DISPLAY EPS, alias for SHOW EPS command [1 - 90](#)

DISPLAY INPROCESSOR, alias for SHOW
INPROCESSOR command [1 - 93](#)

DISPLAY JOB, alias for SHOW JOB command [1 - 96](#)

DISPLAY LOG, alias for SHOW LOG command
[1 - 99](#)

DISPLAY OUTPROCESSOR, alias for SHOW
OUTPROCESSOR command [1 - 101](#)

DISPLAY QUEUE, alias for SHOW QUEUE
command [1 - 103](#)

DISPLAY SPOOL, alias for SHOW SPOOL
command [1 - 106](#)

DO, alias for EXECUTE command [1 - 10](#)

ERASE, alias for CLEAR command [1 - 5](#)

HLQ, alias for HIGHLEVELQUALIFIER
parameter of SET SPOOL command [1 - 83](#)

HOSTNAME, alias for LOCALHOSTNAME
parameter of SET QUEUE command [1 - 74](#)

INCLUDE, alias for EXECUTE command [1 - 10](#)

JESACB, alias for JESAPPLID parameter of
SET INPROCESSOR command [1 - 41](#)

JESPSWD, alias for JESPASSWORD
parameter of SET INPROCESSOR
command [1 - 42](#)

JOBPGMR, alias for JOBPROGRAMMER
parameter of SET INPROCESSOR
command [1 - 37](#)

LINECT
alias for LINECOUNT parameter of SET
INPROCESSOR command [1 - 38](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- alias for LINECOUNT parameter of SET JOB command [1 - 50](#)
- LIST EPS, alias for SHOW EPS command [1 - 90](#)
- LIST INPROCESSOR, alias for SHOW INPROCESSOR command [1 - 93](#)
- LIST JOB, alias for SHOW JOB command [1 - 96](#)
- LIST LOG, alias for SHOW LOG command [1 - 99](#)
- LIST OUTPROCESSOR, alias for SHOW OUTPROCESSOR command [1 - 101](#)
- LIST QUEUE, alias for SHOW QUEUE command [1 - 103](#)
- LIST SPOOL, alias for SHOW SPOOL command [1 - 106](#)
- LPORTS, alias for LOCALPORTS parameter of SET QUEUE command [1 - 75](#)
- LUNAME
 - alias for LOGICALUNITNAME parameter of START command [1 - 110](#), [1 - 111](#)
- LUTABLE, alias for LOGICALUNITTABLE parameter of SET INPROCESSOR command [1 - 46](#)
- MGMTCLAS, alias for MANAGEMENTCALSS parameter of SET SPOOL command [1 - 83](#)
- MODIFY EPS, alias for SET EPS command [1 - 28](#)
- MODIFY INPROCESSOR, alias for SET INPROCESSOR command [1 - 31](#)
- MODIFY JOB, alias for SET JOB command [1 - 48](#)
- MODIFY LOG, alias for SET LOG command [1 - 52](#)
- MODIFY OUTPROCESSOR, alias for SET OUTPROCESSOR command [1 - 56](#)
- MODIFY QUEUE, alias for SET QUEUE command [1 - 62](#)
- MODIFY SPOOL, alias for SET SPOOL command [1 - 81](#)
- MSGLEVEL
 - alias for MESSAGELEVEL parameter of SET EPS command [1 - 29](#)
 - alias for MESSAGELEVEL parameter of SET INPROCESSOR command [1 - 20](#), [1 - 25](#), [1 - 33](#)
 - alias for MESSAGELEVEL parameter of SET LOG command [1 - 54](#)
 - alias for MESSAGELEVEL parameter of SET OUTPROCESSOR command [1 - 58](#)
 - alias for MESSAGELEVEL parameter of SET QUEUE command [1 - 66](#)
- NJEACB, alias for NJEAPPLID parameter of SET INPROCESSOR command [1 - 42](#)
- NJEPSWD, alias for NJEPASSWORD parameter of SET INPROCESSOR command [1 - 43](#)
- PERFORM, alias for EXECUTE command [1 - 10](#)
- QUERY EPS, alias for SHOW EPS command [1 - 90](#)
- QUERY INPROCESSOR, alias for SHOW INPROCESSOR command [1 - 93](#)
- QUERY JOB, alias for SHOW JOB command [1 - 96](#)
- QUERY LOG, alias for SHOW LOG command [1 - 99](#)
- QUERY OUTPROCESSOR, alias for SHOW OUTPROCESSOR command [1 - 101](#)
- QUERY QUEUE, alias for QUERY QUEUE command [1 - 103](#)
- QUERY SPOOL, alias for SHOW SPOOL command [1 - 106](#)
- RCNT, alias for RETRYCOUNT parameter of SET QUEUE command [1 - 67](#)
- RHOST, alias for REMOTEHOST parameter of SET QUEUE command [1 - 76](#), [1 - 79](#)
- RINT, alias for RETRYINTERVAL parameter of SET QUEUE command [1 - 67](#)
- RPORT, alias for REMOTEPORT parameter of SET QUEUE command [1 - 76](#), [1 - 79](#)
- RPRINTER
 - alias for REMOTEPRINTER parameter of SET QUEUE command [1 - 77](#)
- RQUEUE, alias for REMOTEPRINTER parameter of SET QUEUE command [1 - 77](#)
- STORCLAS, alias for STORAGECLASS parameter of SET SPOOL command [1 - 84](#)
- ALL parameter
 - SHOW ALTERNATECONSOLE command [1 - 86](#), [1 - 88](#)
 - SHOW EPS command [1 - 90](#)
 - SHOW INPROCESSOR command [1 - 93](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

SHOW JOB command [1 - 96](#)
SHOW LOG command [1 - 99](#)
SHOW OUTPROCESSOR command [1 - 101](#)
SHOW QUEUE command [1 - 103](#)
SHOW SPOOL command [1 - 106](#)
ALLJOBS parameter, SHOW SPOOL command [1 - 106](#)
ALLOCATION parameter, SET SPOOL command [1 - 81](#)
ALTCONS, alias for ALTERNATECONSOLE
 command [1 - 6](#), [1 - 18](#), [1 - 23](#), [1 - 86](#), [1 - 88](#)
ascii object fields table
 JES-to-ASCII converter exit [A - 46](#)
 LU1 converter exit [A - 38](#)
Assembler interface for Dataset User Exit [A - 17](#)
assign numbers to strings, in Dataset User Exit [B - 9](#)
assignment statement type, User Exit Language [B - 8](#)
ATTRIBUTES parameter, SHOW JOB command [1 - 96](#)
AVERAGERECORDS parameter, SET SPOOL
 command [1 - 82](#)
AVGREC, alias for AVERAGERECORDS
 parameter of SET SPOOL command [1 - 82](#)

B

banner page
 determining beginning and end of SOLVE:EPS
 job [A - 29](#)
 determining when to insert what data [A - 31](#)
 including multiple pieces of information in
 returnstring text [A - 30](#)
 inserting block letters [A - 29](#)
BANNER parameter, SET QUEUE command [1 - 70](#)
beginning and end of SOLVE:EPS job, determining
 [A - 29](#)
Binary converter exit [A - 41](#)
 object fields table [A - 42](#)
 rc object fields table [A - 55](#)
 return object fields table [A - 55](#)
bit literals, User Exit Language data category [B - 4](#)
bits, User Exit Language data type [B - 2](#)
block letters, inserting in Dataset User Exit [A - 29](#)

BLOCKSIZE parameter, SET SPOOL command [1 - 82](#)
boolean object fields table, Dispatch User Exit [A - 10](#)
break, User Exit Language statement type [B - 9](#)
BSD-specific parameters
 SET INPROCESSOR command [1 - 40](#)
 NOSECURITY [1 - 18](#), [1 - 40](#)
 PORT [1 - 20](#), [1 - 40](#)
 SECURITY [1 - 18](#), [1 - 40](#)
 SYSID [1 - 21](#), [1 - 26](#), [1 - 40](#)
 SET OUTPROCESSOR command [1 - 60](#)
 SYSID [1 - 60](#)
 SET QUEUE command [1 - 70](#)
 BANNER [1 - 70](#)
 CONTROLFILE [1 - 70](#)
 CONTROLSTRINGPREFIX [1 - 71](#), [1 - 78](#)
 CONTROLSTRINGSUFFIX [1 - 72](#), [1 - 78](#)
 FILTER [1 - 73](#)
 LOCALHOSTNAME [1 - 74](#)
 LOCALPORTS [1 - 75](#)
 NOBANNER [1 - 70](#)
 NOCONTROLSTRINGPREFIX [1 - 71](#), [1 - 78](#)
 NOCONTROLSTRINGSUFFIX [1 - 72](#), [1 - 78](#)
 REMOTEHOST [1 - 76](#), [1 - 79](#)
 REMOTEPORT [1 - 76](#), [1 - 79](#)
 REMOTEPRINTER [1 - 77](#)

C

CFILE, alias for CONTROLFILE parameter of SET
 QUEUE command [1 - 70](#)
changing a job's dispatching priority, Dispatch User
 Exit [A - 14](#)
changing control strings, Dispatch User Exit [A - 12](#)
character handling, LU1
 overriding [A - 58](#)
 selective [A - 61](#)
CHARx parameter
 SET INPROCESSOR command [1 - 36](#)
 SET JOB command [1 - 49](#)
checkpoint conversion utility [2 - 3](#)
 level 1 to level 2 [2 - 4](#)
 running [2 - 4](#)
 specifying data translation in job records [2 - 4](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- UCS field [2 - 4](#)
- CLASS parameter
 - HOLD command [1 - 11](#)
 - PURGE command [1 - 13](#)
 - RELEASE command [1 - 15](#)
 - SET INPROCESSOR command [1 - 36](#)
 - SET JOB command [1 - 49](#)
 - SET LOG command [1 - 52](#)
- CLEAR command [1 - 5](#)
 - MODEL parameter [1 - 5](#)
- command Body [1 - 3](#)
- commands [1 - 6](#)
 - CLEAR [1 - 5](#)
 - DEFINE CONVERTER [1 - 6](#)
 - DEFINE INPROCESSOR [1 - 7](#)
 - DEFINE OUTPROCESSOR [1 - 8](#)
 - DEFINE QUEUE [1 - 9](#)
 - entering [1 - 2](#)
 - EXECUTE [1 - 10](#)
 - HOLD [1 - 11](#)
 - PROMPT [1 - 11](#)
 - PURGE [1 - 13](#)
 - REFRESH [1 - 14](#)
 - RELEASE [1 - 15](#)
 - RESET [1 - 16](#)
 - ROUTE [1 - 17](#)
 - SET ALTERNATECONSOLE [1 - 18, 1 - 23](#)
 - SET CONVERTER [1 - 27](#)
 - SET EPS [1 - 28](#)
 - SET INPROCESSOR [1 - 31](#)
 - SET JOB [1 - 48](#)
 - SET LOG [1 - 52](#)
 - SET OUTPROCESSOR [1 - 56](#)
 - SET QUEUE [1 - 62](#)
 - SET SPOOL [1 - 81](#)
 - SHOW ALTERNATECONSOLE [1 - 86, 1 - 88](#)
 - SHOW CONVERTER [1 - 89](#)
 - SHOW EPS [1 - 90](#)
 - SHOW INPROCESSOR [1 - 93](#)
 - SHOW JOB [1 - 96](#)
 - SHOW LOG [1 - 99](#)
 - SHOW OUTPROCESSOR [1 - 101](#)
 - SHOW QUEUE [1 - 103](#)
 - SHOW SPOOL [1 - 106](#)
 - SHUTDOWN [1 - 109](#)
 - SPIN [1 - 109](#)
 - START [1 - 110](#)
 - STOP [1 - 111](#)
- comments statement type, User Exit Language [B - 9](#)
- conditional statement type, User Exit Language [B - 6](#)
- CONSOLE parameter
 - SHOW ALTERNATECONSOLE command [1 - 86, 1 - 88](#)
 - SHOW EPS command [1 - 90](#)
 - SHOW INPROCESSOR command [1 - 93](#)
 - SHOW JOB command [1 - 96](#)
 - SHOW LOG command [1 - 99](#)
 - SHOW OUTPROCESSOR command [1 - 101](#)
 - SHOW QUEUE command [1 - 103](#)
 - SHOW SPOOL command [1 - 106](#)
- CONTROLFILE parameter, SET QUEUE command [1 - 70](#)
- CONTROLLINEEXIT parameter, SET QUEUE command [1 - 71](#)
- CONTROLSTRING parameter, REFRESH command [1 - 14](#)
- CONTROLSTRINGPREFIX parameter
 - SET QUEUE command [1 - 71, 1 - 78](#)
- CONTROLSTRINGSUFFIX parameter
 - SET QUEUE command [1 - 72, 1 - 78](#)
- converter exits
 - Binary [A - 41](#)
 - JES-to-ASCII exit [A - 43](#)
 - LU1 [A - 34](#)
 - Translate/Only [A - 40](#)
- Converter Initialization User Exit
 - ascii object fields table [A - 38](#)
 - binary object fields table [A - 42](#)
 - ebcdic object fields table [A - 37](#)
 - horizontal format object fields table [A - 38](#)
 - JES-to-ASCII ascii object fields table [A - 46](#)
 - JES-to-ASCII ebcdic object fields table [A - 46](#)
 - JES-to-ASCII object fields table [A - 44](#)
 - JES-to-ASCII options object fields table [A - 45](#)
 - object fields table [A - 35](#)
 - option fields table [A - 36](#)
 - scscharacter object fields table [A - 39](#)
 - scstype object fields table [A - 40](#)
 - Translate/Only object fields table [A - 42](#)
 - vertical format object fields table [A - 39](#)
- converter object fields table [A - 54](#)
 - Binary converter exit [A - 42](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Translate/Only converter exit [A - 42](#)
converter object name table, CONVERTSELECTOR
exit [A - 33](#)
CONVERTSELECTOR exit
 converter object name field table [A - 33](#)
 object fields table [A - 32](#)
 Outprocessor User Exit [A - 1](#), [A - 32](#)
 rc object fields table [A - 33](#)
 return object fields table [A - 33](#)
CONVERTSELECTOR parameter, SET
 OUTPROCESSOR command [1 - 56](#)
COUNTERS parameter
 SHOW EPS command [1 - 90](#)
 SHOW INPROCESSOR command [1 - 93](#)
 SHOW JOB command [1 - 96](#)
 SHOW LOG command [1 - 99](#)
 SHOW OUTPROCESSOR command [1 - 101](#)
 SHOW QUEUE command [1 - 103](#)
 SHOW SPOOL command [1 - 106](#)
CREATE INPROCESSOR, alias for DEFINE
 INPROCESSOR command [1 - 7](#)
CREATE OUTPROCESSOR, alias for DEFINE
 OUTPROCESSOR command [1 - 8](#)
CREATE QUEUE, alias for DEFINE QUEUE
 command [1 - 9](#)
CTLFILE, alias for CONTROLFILE parameter of
 SET QUEUE command [1 - 70](#)
CTLSTR, alias for CONTROLSTRING parameter of
 REFRESH command [1 - 14](#)
CTLSTRPRE, alias for CONTROLSTRINGPREFIX
 parameter of SET QUEUE command [1 - 71](#),
 [1 - 78](#)
CTLSTRSUF, alias for CONTROLSTRINGSUFFIX
 parameter of SET QUEUE command [1 - 72](#),
 [1 - 78](#)

D

data categories
 User Exit Language
 bit literals [B - 4](#)
 numeric literals [B - 3](#)
 objects with fields [B - 3](#)
 string literals [B - 3](#)

data objects table, Dispatch User Exit [A - 7](#)
data type field, querying in Dataset User Exit [A - 30](#)
data types
 User Exit Language numbers [B - 2](#)
 User Exit Language, bits [B - 2](#)
 User Exit Language, strings [B - 2](#)
DATACLASS parameter, SET SPOOL command [1 - 82](#)
Dataset User Exit
 Assembler exit, new keyword [A - 15](#)
 Assembler interface [A - 17](#)
 Assembler, new keyword [A - 15](#)
 assignment of numbers to strings [B - 9](#)
 beginning and end of SOLVE:EPS job [A - 29](#)
 boolean object fields table [A - 24](#)
 call object fields table [A - 22](#)
 calling [A - 28](#)
 calltype object fields table [A - 24](#)
 data objects table [A - 20](#)
 dataset objects table [A - 21](#)
 datasubtype object fields table [A - 27](#)
 datatype object fields table [A - 26](#)
 determining when to insert what data [A - 31](#)
 entry [A - 17](#)
 hexadecimal string literals [B - 4](#)
 including multiple pieces of information in
 returnstring text [A - 30](#)
 inserting block letters [A - 29](#)
 interface [A - 19](#)
 job object fields table [A - 22](#)
 jobdisposition object fields table [A - 25](#)
 leading zeroes [A - 31](#)
 multiple pieces of information in returnstring text
 [A - 30](#)
 querying data type and subtype fields [A - 30](#)
 querying disposition field [A - 29](#)
 queue object fields table [A - 21](#)
 rc object fields table [A - 24](#)
 return object fields table [A - 25](#)
 return table [A - 18](#)
 returnstring object fields table [A - 24](#)
 sample [A - 28](#)
 setting multiple returnstring options [A - 28](#)
 specifying [A - 15](#)
 stringoptions object fields table [A - 26](#)
 time object fields table [A - 25](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

time structure object fields table [A - 19](#)
DATASETEXIT parameter
 Assembler Dataset User Exits [A - 15](#)
 assign a Dataset User Exit to a queue [A - 16](#)
DBCS
 SOSI handling options [3 - 2](#), [3 - 5](#)
dbcssupport object fields table [A - 56](#)
Debug, SOLVE:EPS-Admin parameter [C - 7](#)
DEFAULT parameter
 SET INPROCESSOR command [1 - 31](#)
 SET OUTPROCESSOR command [1 - 57](#)
 SET QUEUE command [1 - 63](#)
DEFINE ALTERNATECONSOLE [1 - 6](#)
DEFINE ALTERNATECONSOLE command [1 - 6](#)
 TYPE parameter [1 - 6](#)
DEFINE CONVERTER command [1 - 6](#)
DEFINE INPROCESSOR command [1 - 7](#)
 LIKE parameter [1 - 7](#)
 MODEL parameter [1 - 7](#)
 TYPE parameter [1 - 7](#)
DEFINE OUTPROCESSOR command [1 - 8](#)
 LIKE parameter [1 - 8](#)
 MODEL parameter [1 - 8](#)
 TYPE parameter [1 - 8](#)
DEFINE QUEUE command [1 - 9](#)
 LIKE parameter [1 - 9](#)
 MODEL parameter [1 - 9](#)
DESCRIPTION parameter, SET QUEUE command [1 - 63](#)
DESTINATION parameter
 HOLD command [1 - 11](#)
 PURGE command [1 - 13](#)
 RELEASE command [1 - 15](#)
 SET INPROCESSOR command [1 - 36](#)
 SET JOB command [1 - 50](#)
 SET LOG command [1 - 52](#)
Dispatch User Exit [A - 6](#)
 boolean object fields table [A - 10](#)
 data objects table [A - 7](#)
 EPSUSEX user exit language [A - 6](#)
 holding a job [A - 13](#)
 job object fields table [A - 9](#)
 message object fields table [A - 12](#)
 queue object fields table [A - 8](#)
 rc object fields table [A - 11](#)
 return object fields table [A - 11](#)

SET EPS command, TRACEWORD parameter [1 - 30](#)
SET QUEUE command
 DESCRIPTION parameter [1 - 63](#)
 DISPATCHEXIT parameter [1 - 63](#)
specifying [A - 6](#)
tips [A - 12](#), [A - 13](#)
 changing a job's dispatching priority [A - 14](#)
 changing control strings [A - 12](#)
 requeuing a job [A - 13](#)
DISPATCHEXIT parameter, SET QUEUE command [1 - 63](#)
DISPLAY EPS, alias for SHOW EPS command [1 - 90](#)
DISPLAY INPROCESSOR, alias for SHOW INPROCESSOR command [1 - 93](#)
DISPLAY JOB, alias for SHOW JOB command [1 - 96](#)
DISPLAY LOG, alias for SHOW LOG command [1 - 99](#)
DISPLAY OUTPROCESSOR, alias for SHOW OUTPROCESSOR command [1 - 101](#)
DISPLAY QUEUE, alias for SHOW QUEUE command [1 - 103](#)
DISPLAY SPOOL, alias for SHOW SPOOL command [1 - 106](#)
disposition field, querying in Dataset User Exit [A - 29](#)
DISPOSITION parameter
 SET JOB command [1 - 48](#)
 SET QUEUE command [1 - 64](#)
DO, alias for EXECUTE command [1 - 10](#)
DRAIN parameter
 SHUTDOWN command [1 - 109](#)
 STOP command [1 - 111](#)
Dummy Outprocessor, SOLVE:EPS-Admin [C - 8](#)

E

ebcdic object fields table
 JES-to-ASCII converter exit [A - 46](#)
 LU1 converter exit [A - 37](#)
end-of-message, LU1 converter, ignoring [A - 60](#)
ENVIRONMENTMEMBER parameter

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

SET ALTERNATECONSOLE command [1 - 18](#), [1 - 23](#)
SET INPROCESSOR command [1 - 32](#), [1 - 57](#)
EPSUSEX
 Dataset User Exit language [A - 15](#)
 Dispatch User Exit [A - 6](#)
 interface [A - 19](#)
 basic processing [A - 19](#)
 keywords [B - 10](#)
 tips [A - 28](#)
 User Exit Language for Dispatch User Exit [B - 2](#)
ERASE, alias for CLEAR command [1 - 5](#)
error messages, types of [2 - 12](#)
/etc/printcap [3 - 6](#)
EVERY parameter
 SHOW ALTERNATECONSOLE command [1 - 86](#), [1 - 88](#)
 SHOW EPS command [1 - 90](#)
 SHOW INPROCESSOR command [1 - 93](#)
 SHOW JOB command [1 - 96](#)
 SHOW LOG command [1 - 99](#)
 SHOW OUTPROCESSOR command [1 - 101](#)
 SHOW QUEUE command [1 - 103](#)
 SHOW SPOOL command [1 - 106](#)
EXECUTE command [1 - 10](#)
 SOLVE:EPS-Admin [C - 4](#)
extended class parameter
 format for LPD server [3 - 8](#)

F

FCB parameter
 SET INPROCESSOR command [1 - 37](#)
 SET JOB command [1 - 50](#)
FILENAME_TIMESTAMP parameter, SET QUEUE
 command [1 - 73](#)
FILTER parameter
 SET QUEUE command [1 - 73](#)
FLASH parameter
 SET INPROCESSOR command [1 - 37](#)
 SET JOB command [1 - 50](#)
FORCE parameter
 ROUTE command [1 - 17](#)
 SHUTDOWN command [1 - 109](#)

STOP command [1 - 111](#)
FORMDEF parameter
 SET INPROCESSOR command [1 - 37](#)
 SET JOB command [1 - 50](#)
formfeed object fields table [A - 57](#)
FORMS parameter
 HOLD command [1 - 11](#)
 PURGE command [1 - 13](#)
 RELEASE command [1 - 15](#)
 SET INPROCESSOR command [1 - 37](#)
 SET JOB command [1 - 50](#)
 SET LOG command [1 - 53](#)

H

hexadecimal string literals, in Dataset User Exit [B - 4](#)
HIGHLEVELQUALIFIER parameter, SET SPOOL
 command [1 - 83](#)
HLQ, alias for HIGHLEVELQUALIFIER parameter of
 SET SPOOL command [1 - 83](#)
HOLD command [1 - 11](#)
 CLASS parameter [1 - 11](#)
 DESTINATION parameter [1 - 11](#)
 FORMS parameter [1 - 11](#)
 ORPHANS parameter [1 - 11](#)
 QUEUE parameter [1 - 11](#)
 THRU parameter [1 - 11](#)
 USER parameter [1 - 11](#)
HOLD parameter
 RESET command [1 - 16](#)
 SET INPROCESSOR command [1 - 37](#)
 SET JOB command [1 - 50](#)
holding a job, Dispatch User Exit [A - 13](#)
horizontal format object fields table, LU1 converter
 exit [A - 38](#)
HOSTNAME, alias for LOCALHOSTNAME
 parameter of SET QUEUE command [1 - 74](#)

I

IMMEDIATE parameter
 SHUTDOWN command [1 - 109](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

STOP command [1 - 111](#)
Inactivity Timeout, SOLVE:ESP-Admin parameter [C - 6](#)
INCLUDE, alias for EXECUTE command [1 - 10](#)
INDENT parameter
 SET INPROCESSOR command [1 - 37](#)
 SET JOB command [1 - 50](#)
 SET QUEUE command [1 - 73](#)
informational messages, types of [2 - 12](#)
Initialization User Exits, converter [A - 34](#)
 Binary [A - 41](#)
 JES-to-ASCII converter [A - 43](#)
 LU1 converter [A - 34](#)
 Translate/Only [A - 40](#)
Inprocessor
 SOLVE:EPS-Admin [C - 11](#)
 Zoom [C - 11](#)
INPROCESSOR parameter, SHOW EPS command [1 - 90](#)
input sources [1 - 2](#)

J

JCL
 sample, T07SCVCH [2 - 5](#)
 statements [2 - 2](#)
JES2
 printing on [3 - 5](#)
JES3
 printing on [3 - 5](#)
JESACB, alias for JESAPPLID parameter of SET INPROCESSOR command [1 - 41](#)
JESAPPLID parameter, SET INPROCESSOR command [1 - 41](#)
JESNODE parameter, SET INPROCESSOR command [1 - 41](#)
JESPASSWORD parameter, SET INPROCESSOR command [1 - 42](#)
JESPSWD, alias for JESPASSWORD parameter of SET INPROCESSOR command [1 - 42](#)
JES-to-ASCII converter exit [A - 43](#)
 ascii object fields table [A - 46](#)
 ebcdic object fields table [A - 46](#)
 object fields table [A - 44](#)

options object fields table [A - 45](#)
rc object fields table [A - 55](#)
return object fields table [A - 55](#)
job defaults parameters
SET INPROCESSOR command [1 - 36](#), [1 - 49](#)
 ACCOUNT [1 - 36](#)
 CHARx [1 - 36](#)
 CLASS [1 - 36](#)
 DESTINATION [1 - 36](#)
 FCB [1 - 37](#)
 FLASH [1 - 37](#)
 FORMDEF [1 - 37](#)
 FORMS [1 - 37](#)
 HOLD [1 - 37](#)
 INDENT [1 - 37](#)
 JOBNAME [1 - 37](#)
 JOBPROGRAMMER [1 - 37](#)
 JOBROOM [1 - 37](#)
 LINECOUNT [1 - 38](#)
 NOACCOUNT [1 - 36](#)
 NOCHARx [1 - 36](#)
 NOCLASS [1 - 36](#)
 NODESTINATION [1 - 36](#)
 NOFCB [1 - 37](#)
 NOFLASH [1 - 37](#)
 NOFORMDEF [1 - 37](#)
 NOFORMS [1 - 37](#)
 NOHOLD [1 - 37](#)
 NOINDENT [1 - 37](#)
 NOJOBNAME [1 - 37](#)
 NOJOBPROGRAMMER [1 - 37](#)
 NOJOBROOM [1 - 37](#)
 NOLINECOUNT [1 - 38](#)
 NONOTIFYUSER [1 - 38](#)
 NOPAGEDEF [1 - 38](#)
 NOPRMODE [1 - 38](#)
 NOTIFYUSER [1 - 38](#)
 NOTRC [1 - 38](#)
 NOUCS [1 - 38](#)
 NOWIDTH [1 - 38](#)
 NOWRITER [1 - 39](#)
 PAGEDEF [1 - 38](#)
 PRMODE [1 - 38](#)
 TRC [1 - 38](#)
 UCS [1 - 38](#)
 WIDTH [1 - 38](#)
 WRITER [1 - 39](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

SET JOB command

CHARx [1 - 49](#)
CLASS [1 - 49](#)
DESTINATION [1 - 50](#)
FCB [1 - 50](#)
FLASH [1 - 50](#)
FORMDEF [1 - 50](#)
FORMS [1 - 50](#)
HOLD [1 - 50](#)
INDENT [1 - 50](#)
LINECOUNT [1 - 50](#)
NOCHARx [1 - 49](#)
NOCLASS [1 - 49](#)
NODESTINATION [1 - 50](#)
NOFCB [1 - 50](#)
NOFLASH [1 - 50](#)
NOFORMDEF [1 - 50](#)
NOFORMS [1 - 50](#)
NOHOLD [1 - 50](#)
NOINDENT [1 - 50](#)
NOLINECOUNT [1 - 50](#)
NOPAGEDEF [1 - 51](#)
NOPRMODE [1 - 51](#)
NOTRC [1 - 51](#)
NOUCS [1 - 51](#)
NOWIDTH [1 - 51](#)
NOWRITER [1 - 51](#)
PAGEDEF [1 - 51](#)
PRMODE [1 - 51](#)
TRC [1 - 51](#)
UCS [1 - 51](#)
WIDTH [1 - 51](#)
WRITER [1 - 51](#)

Job Filters, SOLVE:EPS-Admin Jobs [C - 10](#)

job object fields table, Dispatch User Exit [A - 9](#)

JOBNAME parameter, SET INPROCESSOR command [1 - 37](#)

JOBPGMR, alias for JOBPROGRAMMER parameter of SET INPROCESSOR command [1 - 37](#)

JOBPROGRAMMER parameter, SET INPROCESSOR command [1 - 37](#)

JOBROOM parameter, SET INPROCESSOR command [1 - 37](#)

JOBS parameter
SHOW OUTPROCESSOR command [1 - 101](#)

SHOW QUEUE command [1 - 103](#)

SHOW SPOOL command [1 - 106](#)

Jobs, SOLVE:EPS-Admin [C - 10](#)

L

language object fields table [A - 54](#)

leading zeroes, in Dataset User Exit [A - 31](#)

LIKE parameter

DEFINE INPROCESSOR command [1 - 7](#)

DEFINE OUTPROCESSOR command [1 - 8](#)

DEFINE QUEUE command [1 - 9](#)

LINECOUNT parameter

SET INPROCESSOR command [1 - 38](#)

SET JOB command [1 - 50](#)

LINECT

alias for LINECOUNT parameter of SET INPROCESSOR command [1 - 38](#)

alias for LINECOUNT parameter of SET JOB command [1 - 50](#)

LINESIZE parameter, SET QUEUE command [1 - 64](#)

LIST EPS, alias for SHOW EPS command [1 - 90](#)

LIST INPROCESSOR, alias for SHOW INPROCESSOR command [1 - 93](#)

LIST JOB, alias for SHOW JOB command [1 - 96](#)

LIST LOG, alias for SHOW LOG command [1 - 99](#)

LIST OUTPROCESSOR, alias for SHOW OUTPROCESSOR command [1 - 101](#)

LIST QUEUE, alias for SHOW QUEUE command [1 - 103](#)

LIST SPOOL, alias for SHOW SPOOL command [1 - 106](#)

LOCALDELAYPORTTIMER parameter, SET QUEUE command [1 - 74](#)

LOCALHOSTNAME parameter, SET QUEUE command [1 - 74](#)

LOCALPORTS parameter, SET QUEUE command [1 - 75](#)

LOG parameter

SHOW ALTERNATECONSOLE command [1 - 86](#), [1 - 88](#)

SHOW EPS command [1 - 90](#)

SHOW INPROCESSOR command [1 - 93](#)

SHOW JOB command [1 - 96](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

SHOW LOG command [1 - 99](#)
SHOW OUTPROCESSOR command [1 - 101](#)
SHOW QUEUE command [1 - 103](#)
SHOW SPOOL command [1 - 106](#)
LOGICALUNITNAME parameter, START command
[1 - 110](#), [1 - 111](#)
LOGICALUNITTABLE parameter, SET
INPROCESSOR command [1 - 46](#)
LOGLEVEL parameter
SET EPS command [1 - 28](#)
SET INPROCESSOR command [1 - 19](#), [1 - 24](#), [1 - 32](#)
SET LOG command [1 - 53](#)
SET OUTPROCESSOR command [1 - 58](#)
SET QUEUE command [1 - 65](#)
Logs, SOLVE:EPS-Admin [C - 11](#)
LPORTS, alias for LOCALPORTS parameter of SET
QUEUE command [1 - 75](#)
lpr
UNIX print command [3 - 5](#)
usage example [3 - 10](#)
LU1 character handling, overriding [A - 58](#)
LU1 converter
default [A - 58](#)
defining [A - 59](#)
escape sequences in data [A - 60](#)
ignoring end-of-message [A - 60](#)
selective character handling [A - 61](#)
LU1 converter exit
ascii object fields table [A - 38](#)
ebcdic object fields table [A - 37](#)
horizontal format object fields table [A - 38](#)
Initialization User Exit [A - 34](#)
object fields [A - 36](#)
object fields table [A - 35](#)
options table [A - 36](#)
rc object fields table [A - 55](#)
return object fields table [A - 55](#)
scscharacter object fields table [A - 39](#)
scstype object fields [A - 40](#)
vertical format object fields table [A - 39](#)
LU3 printers [3 - 4](#)
LUNAME
alias for LOGICALUNITNAME parameter of
START command [1 - 110](#), [1 - 111](#)

LUTABLE, alias for LOGICALUNITTABLE
parameter of SET INPROCESSOR
command [1 - 46](#)

M

MANAGEMENTCLASS parameter, SET SPOOL
command [1 - 83](#)
message object fields table, for Dispatch User Exit [A - 12](#)
MESSAGELEVEL parameter
SET EPS command [1 - 29](#)
SET INPROCESSOR command [1 - 20](#), [1 - 25](#), [1 - 33](#)
SET LOG command [1 - 54](#)
SET OUTPROCESSOR command [1 - 58](#)
SET QUEUE command [1 - 66](#)
MGMTCLAS, alias for MANAGEMENTCLASS
parameter of SET SPOOL command [1 - 83](#)
MODEL parameter
CLEAR command [1 - 5](#)
DEFINE INPROCESSOR command [1 - 7](#)
DEFINE OUTPROCESSOR command [1 - 8](#)
DEFINE QUEUE command [1 - 9](#)
SHOW INPROCESSOR command [1 - 93](#)
SHOW OUTPROCESSOR command [1 - 101](#)
SHOW QUEUE command [1 - 103](#)
MODEL parameter, SHOW EPS command [1 - 90](#)
MODIFY EPS, alias for SET EPS command [1 - 28](#)
MODIFY INPROCESSOR, alias for SET
INPROCESSOR command [1 - 31](#)
MODIFY JOB, alias for SET JOB command [1 - 48](#)
MODIFY LOG, alias for SET LOG command [1 - 52](#)
MODIFY OUTPROCESSOR, alias for SET
OUTPROCESSOR command [1 - 56](#)
MODIFY QUEUE, alias for SET QUEUE command [1 - 62](#)
MODIFY SPOOL, alias for SET SPOOL command [1 - 81](#)
MSGLEVEL
alias for MESSAGELEVEL parameter of SET
EPS command [1 - 29](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

alias for MESSAGELEVEL parameter of SET INPROCESSOR command [1 - 20](#), [1 - 25](#), [1 - 33](#)
alias for MESSAGELEVEL parameter of SET LOG command [1 - 54](#)
alias for MESSAGELEVEL parameter of SET OUTPROCESSOR command [1 - 58](#)
alias for MESSAGELEVEL parameter of SET QUEUE command [1 - 66](#)
multiple returnstring options, setting in Dataset User Exit [A - 28](#)

N

Native Interface Modeling, SOLVE:ESP-Admin parameter [C - 7](#)
New Queue, SOLVE:EPS-Admin queue [C - 9](#)
NEWQUEUE command, SOLVE:EPS-Admin [C - 4](#)
NJE parameter, START command [1 - 110](#), [1 - 111](#)
NJEACB, alias for NJEAPPLID parameter of SET INPROCESSOR command [1 - 42](#)
NJEAPPLID parameter, SET INPROCESSOR command [1 - 42](#)
NJEINPROCESSOR parameter, SET OUTPROCESSOR command [1 - 61](#)
NJENODE parameter, SET INPROCESSOR command [1 - 42](#), [1 - 46](#)
NJEPARTNER, SET INPROCESSOR parameter [1 - 43](#)
NJEPASSWORD parameter, SET INPROCESSOR command [1 - 43](#)
NJEPSWD, alias for NJEPASSWORD parameter of SET INPROCESSOR command [1 - 43](#)
NJE-specific parameters
 SET INPROCESSOR command [1 - 41](#)
 JESAPPLID [1 - 41](#)
 JESNODE [1 - 41](#)
 JESPASSWORD [1 - 42](#)
 NJEAPPLID [1 - 42](#)
 NJENODE [1 - 42](#), [1 - 46](#)
 NJEPASSWORD [1 - 43](#)
 NOJESPASSWORD [1 - 42](#)
 NONJEPASSWORD [1 - 43](#)
 SET OUTPROCESSOR command [1 - 61](#)
 NJEINPROCESSOR [1 - 61](#)
NJETABLE parameter, SET INPROCESSOR command [1 - 43](#)
NOACCOUNT parameter, SET INPROCESSOR command [1 - 36](#)
NOBANNER parameter, SET QUEUE command [1 - 70](#)
NOCHARx parameter
 SET INPROCESSOR command [1 - 36](#)
 SET JOB command [1 - 49](#)
NOCLASS parameter
 SET INPROCESSOR command [1 - 36](#)
 SET JOB command [1 - 49](#)
NOCONSOLE parameter
 SHOW OUTPROCESSOR command [1 - 101](#)
 SHOW QUEUE command [1 - 103](#)
 SHOW SPOOL command [1 - 106](#)
NOCONTROLSTRINGPREFIX parameter, SET QUEUE command [1 - 71](#), [1 - 78](#)
NOCONTROLSTRINGPREFIX parameter, SET QUEUE command [1 - 72](#), [1 - 78](#)
NODEFAULT parameter
 SET INPROCESSOR command [1 - 31](#)
 SET OUTPROCESSOR command [1 - 57](#)
 SET QUEUE command [1 - 63](#)
NODESTINATION parameter
 SET INPROCESSOR command [1 - 36](#)
 SET JOB command [1 - 50](#)
 SET LOG command [1 - 52](#)
NODISPOSITION parameter, SET QUEUE command [1 - 64](#)
NOFCB parameter
 SET INPROCESSOR command [1 - 37](#)
 SET JOB command [1 - 50](#)
NOFLASH parameter
 SET INPROCESSOR command [1 - 37](#)
 SET JOB command [1 - 50](#)
NOFORMDEF parameter
 SET INPROCESSOR command [1 - 37](#)
 SET JOB command [1 - 50](#)
NOFORMS parameter
 SET INPROCESSOR command [1 - 37](#)
 SET JOB command [1 - 50](#)
 SET LOG command [1 - 53](#)
NOHOLD parameter

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

SET INPROCESSOR command [1 - 37](#)
SET JOB command [1 - 50](#)
NOINDENT parameter
SET INPROCESSOR command [1 - 37](#)
SET JOB command [1 - 50](#)
SET QUEUE command [1 - 73](#)
NOJESPASSWORD parameter, SET
INPROCESSOR command [1 - 42](#)
NOJOBNAME parameter, SET INPROCESSOR
command [1 - 37](#)
NOJOBPROGRAMMER parameter, SET
INPROCESSOR command [1 - 37](#)
NOJOBROOM parameter, SET INPROCESSOR
command [1 - 37](#)
NOLINECOUNT parameter
SET INPROCESSOR command [1 - 38](#)
SET JOB command [1 - 50](#)
NOLOG parameter
SHOW OUTPROCESSOR command [1 - 101](#)
SHOW QUEUE command [1 - 103](#)
SHOW SPOOL command [1 - 106](#)
NONJEPASSWORD parameter, SET
INPROCESSOR command [1 - 43](#)
NONOTIFYUSER parameter, SET INPROCESSOR
command [1 - 38](#)
NOPAGEDEF parameter
SET INPROCESSOR command [1 - 38](#)
SET JOB command [1 - 51](#)
NOPRMODE parameter
SET INPROCESSOR command [1 - 38](#)
SET JOB command [1 - 51](#)
NOSECURITY parameter, SET INPROCESSOR
command [1 - 40](#)
NOSPINSYNCHRONIZE parameter, SET LOG
command [1 - 55](#)
NOTIFYUSER parameter, SET INPROCESSOR
command [1 - 38](#)
NOTRC parameter
SET INPROCESSOR command [1 - 38](#)
SET JOB command [1 - 51](#)
NOUCS parameter
SET INPROCESSOR command [1 - 38](#)
SET JOB command [1 - 51](#)
NOUPPERCASE parameter
SET EPS command [1 - 30](#)

SET LOG command [1 - 22](#), [1 - 55](#)
NOWIDTH parameter
SET INPROCESSOR command [1 - 38](#)
SET JOB command [1 - 51](#)
SET QUEUE command [1 - 69](#)
NOWRITER parameter
SET INPROCESSOR command [1 - 39](#)
SET JOB command [1 - 51](#)
numbers, User Exit Language data type [B - 2](#)
numeric literals, User Exit Language data category [B](#)
[- 3](#)

O

object fields for Converter Initialization User Exit
ascii table [A - 38](#)
binary table [A - 42](#)
ebcdic table [A - 37](#)
horizontal format table [A - 38](#)
JES-to-ASCII ascii table [A - 46](#)
JES-to-ASCII ebcdic table [A - 46](#)
JES-to-ASCII options table [A - 45](#)
JES-to-ASCII table [A - 44](#)
LU1 table [A - 35](#)
option table [A - 36](#)
scscharacter table [A - 39](#)
scstype table [A - 40](#)
Translate/Only table [A - 42](#)
vertical format table [A - 39](#)
object fields for Dataset User Exit
boolean table [A - 24](#)
call table [A - 22](#)
calltype table [A - 24](#)
dataset table [A - 21](#)
datasubtype table [A - 27](#)
datatype table [A - 26](#)
job table [A - 22](#)
jobdisposition table [A - 25](#)
queue table [A - 21](#)
rc table [A - 24](#)
return table [A - 18](#), [A - 25](#)
returnstring table [A - 24](#)
stringoptions table [A - 26](#)
time structure table [A - 19](#)
time table [A - 25](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

object fields for Dispatch User Exit

- boolean table [A - 10](#)
- data table [A - 7](#)
- job table [A - 9](#)
- message table [A - 12](#)
- queue table [A - 8](#)
- rc table [A - 11](#)
- return table [A - 11](#)

object fields for Outprocessor User Exit

- CONVERTSELECTOR table [A - 32](#)
- name table [A - 33](#)
- rc table [A - 33](#)
- return table [A - 33](#)

object fields table

- JES-to-ASCII converter exit [A - 44](#)
- LU1 converter exit table [A - 35](#)

OBJECTS parameter, SHOW EPS command [1 - 90](#)

- objects with fields, User Exit Language data category [B - 3](#)

OPENJOBLIMIT parameter, SET INPROCESSOR command [1 - 47](#)

option fields table [A - 36](#)

- JES-to-ASCII converter exit [A - 45](#)

OPTIONS command

- SOLVE:EPS-Admin [C - 4](#)

ORPHANS parameter

- HOLD command [1 - 11](#)
- PURGE command [1 - 13](#)
- RELEASE command [1 - 15](#)
- SHOW SPOOL command [1 - 106](#)

Outprocessor

- SOLVE:EPS-Admin
 - Dummy [C - 8](#)
 - Zoom [C - 8](#)

OUTPROCESSOR parameter

- SET QUEUE command [1 - 66](#)
- SHOW EPS command [1 - 91](#)

Outprocessor User Exit

- CONVERTSELECTOR [A - 1](#), [A - 32](#)
- CONVERTSELECTOR object fields table [A - 32](#)
- rc object fields table [A - 33](#)
- return object fields table [A - 33](#)

P

PAGEDEF parameter

- SET INPROCESSOR command [1 - 38](#)
- SET JOB command [1 - 51](#)

PAGESIZE parameter, SET QUEUE command [1 - 67](#)

parameters

- ALL, SHOW ALTERNATECONSOLE command [1 - 86](#), [1 - 88](#)

- BLOCKSIZE, SET SPOOL command [1 - 82](#)

CLASS

- HOLD command [1 - 11](#)
- PURGE command [1 - 13](#)
- RELEASE command [1 - 15](#)

- CONSOLE, SHOW ALTERNATECONSOLE command [1 - 86](#), [1 - 88](#)

- CONTROLLINEEXIT, SET QUEUE command [1 - 71](#)

CONVERTSELECTOR

- SET OUTPROCESSOR command [1 - 56](#)

- DESCRIPTION, SET QUEUE command [1 - 63](#)

DESTINATION

- HOLD command [1 - 11](#)
- PURGE command [1 - 13](#)
- RELEASE command [1 - 15](#)

- DISPATCHEXIT, SET QUEUE command [1 - 63](#)

ENVIRONMENTMEMBER

- SET ALTERNATECONSOLE command [1 - 18](#), [1 - 23](#)

- SET INPROCESSOR command [1 - 32](#), [1 - 57](#)

- EVERY, SHOW ALTERNATECONSOLE command [1 - 86](#), [1 - 88](#)

- FILENAMETIMESTAMP, SET QUEUE command [1 - 73](#)

FORMS

- HOLD command [1 - 11](#)
- PURGE command [1 - 13](#)
- RELEASE command [1 - 15](#)

- HOLD, RESET command [1 - 16](#)

- LOCALDELAYPORTTIMER, SET QUEUE command [1 - 74](#)

- LOG, SHOW ALTERNATECONSOLE command [1 - 86](#), [1 - 88](#)

- MODEL, SHOW EPS command [1 - 90](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

NJEPARTNER, SET INPROCESSOR
command [1 - 43](#)

NJETABLE, SET INPROCESSOR command [1 - 43](#)

OPENJOBLIMIT, SET INPROCESSOR
command [1 - 47](#)

ORPHANS
HOLD command [1 - 11](#)
PURGE command [1 - 13](#)
RELEASE command [1 - 15](#)

PRIORITY, SET INPROCESSOR command [1 - 34](#)

PURGE, RESET command [1 - 16](#)

QUEUE
HOLD command [1 - 11](#)
PURGE command [1 - 13](#)
RELEASE command [1 - 15](#)

READBUFNO, SET SPOOL command [1 - 83](#)

READY, RESET command [1 - 16](#)

SENDALLCOPIES, SET QUEUE command [1 - 68](#)

SESSION, STOP command [1 - 111](#)

SESSIONS
SET ALTERNATECONSOLE command [1 - 21](#)
SET INPROCESSOR command [1 - 34](#)
SET OUTPROCESSOR command [1 - 59](#)
SET QUEUE command [1 - 68](#)

SETTINGS, SHOW ALTERNATECONSOLE
command [1 - 86](#), [1 - 88](#)

SOLVE:EPS-Admin [C - 6](#)

STATUS
PURGE command [1 - 13](#)
RELEASE command [1 - 15](#)
SHOW ALTERNATECONSOLE command [1 - 86](#), [1 - 88](#)

TRACEWORD, SET EPS command [1 - 30](#)

TRANSLATE
SET INPROCESSOR command [1 - 22](#), [1 - 35](#)
SET QUEUE command [1 - 69](#)

UCSTABLE, SET INPROCESSOR command [1 - 44](#), [1 - 45](#)

USER
HOLD command [1 - 11](#)
PURGE command [1 - 13](#)
RELEASE command [1 - 15](#)

WRITEBUFNO, SET SPOOL command [1 - 84](#)

PERFORM, alias for EXECUTE command [1 - 10](#)

PORT parameter, SET INPROCESSOR command [1 - 20](#), [1 - 40](#)

PRIMARYSPACE parameter, SET SPOOL
command [1 - 83](#)

printcap [3 - 6](#)
file sample [3 - 6](#)

PRINTDS, TSO command [3 - 3](#)

printer
virtual, stopping and restarting [2 - 11](#)

printing
on remote printers [3 - 2](#)

PRIORITY parameter
SET INPROCESSOR [1 - 34](#)
SET JOB command [1 - 49](#)

PRMODE parameter
SET INPROCESSOR command [1 - 38](#)
SET JOB command [1 - 51](#)

PROMPT command [1 - 11](#)

PURGE command [1 - 13](#)
CLASS parameter [1 - 13](#)
DESTINATION parameter [1 - 13](#)
FORMS parameter [1 - 13](#)
ORPHANS parameter [1 - 13](#)
QUEUE parameter [1 - 13](#)
STATUS parameter [1 - 13](#)
THRU parameter [1 - 13](#)
USER parameter [1 - 13](#)

PURGE parameter, RESET command [1 - 16](#)

Q

QUERY EPS, alias for SHOW EPS [1 - 90](#)

QUERY INPROCESSOR, alias for SHOW
INPROCESSOR command [1 - 93](#)

QUERY JOB, alias for SHOW JOB command [1 - 96](#)

QUERY LOG, alias for SHOW LOG command [1 - 99](#)

QUERY OUTPROCESSOR, alias for SHOW
OUTPROCESSOR command [1 - 101](#)

QUERY QUEUE, alias for SHOW QUEUE command
[1 - 103](#)

QUERY SPOOL, alias for SHOW SPOOL command
[1 - 106](#)

querying

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

data type and subtype fields in Dataset User Exit
[A - 30](#)
disposition field in Dataset User Exit [A - 29](#)
Queue Filter, SOLVE:EPS-Admin [C - 8](#)
queue object fields
 querying [A - 30](#)
 table for Dispatch User Exit [A - 8](#)
QUEUE parameter
 HOLD command [1 - 11](#)
 PURGE command [1 - 13](#)
 RELEASE command [1 - 15](#)
 SET JOB command [1 - 49](#)
QUEUEDEFAULT parameter, SET INPROCESSOR
 command [1 - 33](#)
Queues and Outprocessors, SOLVE:EPS-Admin [C - 8](#)
QUEUES parameter
 SHOW EPS command [1 - 91](#)
 SHOW OUTPROCESSOR command [1 - 101](#)

R

rc object fields table
 Binary converter exit [A - 55](#)
 CONVERTSELECTOR exit [A - 33](#)
 Dispatch User Exit [A - 11](#)
 JES-to-ASCII converter exit [A - 55](#)
 LU1 converter exit [A - 55](#)
 Translate/Only converter exit [A - 55](#)
RCNT, alias for RETRYCOUNT parameter of SET
 QUEUE command [1 - 67](#)
READBUFNO parameter, SET SPOOL command [1 - 83](#)
READY parameter, RESET command [1 - 16](#)
REFRESH command [1 - 14](#)
 CONTROLSTRING parameter [1 - 14](#)
 SECURITY parameter [1 - 14](#)
 SOLVE:EPS-Admin [C - 4](#)
 SSISELECT parameter [1 - 14](#)
RELEASE command [1 - 15](#)
 CLASS parameter [1 - 15](#)
 DESTINATION parameter [1 - 15](#)
 FORMS parameter [1 - 15](#)
 ORPHANS parameter [1 - 15](#)

 QUEUE parameter [1 - 15](#)
 STATUS parameter [1 - 15](#)
 THRU parameter [1 - 15](#)
 USER parameter [1 - 15](#)
RELOAD command, SOLVE:EPS-Admin [C - 4](#)
Reload Table, SOLVE:EPS_Admin, SOLVE:EPS [C - 12](#)
Remote Machine Name, SOLVE:EPS-Admin
 parameter [C - 6](#)
Remote Password, SOLVE:ESP-Admin parameter
 [C - 6](#)
Remote Port, SOLVE:ESP-Admin parameter [C - 6](#)
Remote User ID, SOLVE:ESP-Admin parameter [C - 6](#)
REMOTEHOST parameter, SET QUEUE command
 [1 - 76](#), [1 - 79](#)
REMOTEPORT parameter, SET QUEUE command
 [1 - 76](#), [1 - 79](#)
REMOTEPRINTER parameter, SET QUEUE
 command [1 - 77](#)
requeuing a job, Dispatch User Exit [A - 13](#)
RESET command [1 - 16](#)
 HOLD parameter [1 - 16](#)
 PURGE parameter [1 - 16](#)
 READY parameter [1 - 16](#)
Reset Job, SOLVE:EPS-Admin Jobs [C - 10](#)
response messages, description [2 - 12](#)
RESPONSEFILTER parameter, SET
 INPROCESSOR command [1 - 21](#), [1 - 24](#)
RETRYCOUNT parameter, SET QUEUE command
 [1 - 67](#)
RETRYINTERVAL parameter, SET QUEUE
 command [1 - 67](#)
return object fields table
 Binary converter exit [A - 55](#)
 CONVERTSELECTOR exit [A - 33](#)
 Dispatch User Exit [A - 11](#)
 JES-to-ASCII converter exit [A - 55](#)
 LU1 converter exit [A - 55](#)
 Translate/Only converter exit [A - 55](#)
returnstring, including multiple pieces of information
 [A - 30](#)
RHOST, alias for REMOTEHOST parameter of SET
 QUEUE command [1 - 76](#), [1 - 79](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

RINT, alias for RETRYINTERVAL parameter of SET QUEUE command [1 - 67](#)
ROUTE command [1 - 17](#)
 FORCE parameter [1 - 17](#)
Route Queue, SOLVE:EPS-Admin queue [C - 9](#)
RPORT, alias for REMOTEPORT parameter of SET QUEUE command [1 - 76](#), [1 - 79](#)
RPRINTER
 alias for REMOTEPRINTER parameter of SET QUEUE command [1 - 77](#)
RQUEUE, alias for REMOTEPRINTER parameter of SET QUEUE command [1 - 77](#)

S

samples

 Dataset User Exit [A - 28](#)
 lpr usage [3 - 10](#)
 printcap file [3 - 6](#)
 SHOW EPS command
 with EVERY and ALL parameters [1 - 91](#)
 with STATUS parameter [1 - 87](#), [1 - 88](#), [1 - 91](#)
 SHOW JOB command with EVERY and ALL parameters [1 - 97](#)
 SHOW LOG command with EVERY and ALL parameters [1 - 99](#)
 SHOW OUTPROCESSOR command with EVERY and ALL parameters [1 - 102](#)
 SHOW QUEUE command with EVERY and ALL parameters [1 - 103](#)
 SHOW SPOOL command with EVERY and ALL parameters [1 - 107](#)
 SOLVE:EPS JCL statements [2 - 2](#)
 T07SCVCH JCL [2 - 5](#)
scscharacter table, LU1 converter exit [A - 39](#)
scstype table, LU1 converter exit [A - 40](#)
SECONDARYSPACE parameter, SET SPOOL command [1 - 84](#)
SECURITY parameter
 REFRESH command [1 - 14](#)
 SET INPROCESSOR command [1 - 40](#)
SENDALLCOPIES parameter, SET QUEUE command [1 - 68](#)
SESSION parameter, STOP command [1 - 111](#)

SESSIONS parameter
 SET ALTERNATECONSOLE command [1 - 21](#)
 SET INPROCESSOR command [1 - 34](#)
 SET OUTPROCESSOR command [1 - 59](#)
 SET QUEUE command [1 - 68](#)
SET ALTERNATECONSOLE command [1 - 18](#), [1 - 23](#)
 ENVIRONMENTMEMBER parameter [1 - 18](#), [1 - 23](#)
SET CONVERTER command [1 - 27](#)
SET EPS command [1 - 28](#)
 LOGLEVEL parameter [1 - 28](#)
 MESSAGELEVEL parameter [1 - 29](#)
 NOUPPERCASE parameter [1 - 30](#)
 TRACEWORD parameter [1 - 30](#)
 UPPERCASE parameter [1 - 30](#)
SET INPROCESSOR command [1 - 31](#)
 BSD-specific parameters [1 - 40](#)
 NOSECURITY [1 - 18](#), [1 - 40](#)
 PORT [1 - 20](#), [1 - 40](#)
 SECURITY [1 - 18](#), [1 - 40](#)
 SYSID [1 - 21](#), [1 - 26](#), [1 - 40](#)
 DEFAULT parameter [1 - 31](#)
 ENVIRONMENTMEMBER parameter [1 - 32](#), [1 - 57](#)
 job defaults parameters [1 - 36](#), [1 - 49](#)
 ACCOUNT [1 - 36](#)
 CHARx [1 - 36](#)
 CLASS [1 - 36](#)
 DESTINATION [1 - 36](#)
 FCB [1 - 37](#)
 FLASH [1 - 37](#)
 FORMDEF [1 - 37](#)
 FORMS [1 - 37](#)
 HOLD [1 - 37](#)
 INDENT [1 - 37](#)
 JOBNAME [1 - 37](#)
 JOBPROGRAMMER [1 - 37](#)
 JOBROOM [1 - 37](#)
 LINECOUNT [1 - 38](#)
 NOACCOUNT [1 - 36](#)
 NOCHARx [1 - 36](#)
 NOCLASS [1 - 36](#)
 NODESTINATION [1 - 36](#)
 NOFCB [1 - 37](#)
 NOFLASH [1 - 37](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

NOFORMDEF 1 - 37
NOFORMS 1 - 37
NOHOLD 1 - 37
NOINDENT 1 - 37
NOJOBNAME 1 - 37
NOJOBPROGRAMMER 1 - 37
NOJOBROOM 1 - 37
NOLINECOUNT 1 - 38
NONOTIFYUSER 1 - 38
NOPAGEDEF 1 - 38
NOPRMODE 1 - 38
NOTIFYUSER 1 - 38
NOTRC 1 - 38
NOUCS 1 - 38
NOWIDTH 1 - 38
NOWRITER 1 - 39
PAGEDEF 1 - 38
PRMODE 1 - 38
TRC 1 - 38
UCS 1 - 38
WIDTH 1 - 38
WRITER 1 - 39
LOGLEVEL parameter 1 - 19, 1 - 24, 1 - 32
MESSAGELEVEL parameter 1 - 20, 1 - 25, 1 - 33
NJEPARTNER parameter 1 - 43
NJE-specific parameters 1 - 41
 JESAPPLID 1 - 41
 JESNODE 1 - 41
 JESPASSWORD 1 - 42
 NJEAPPLID 1 - 42
 NJENODE 1 - 42, 1 - 46
 NJEPASSWORD 1 - 43
 NOJESPASSWORD 1 - 42
 NONJEPASSWORD 1 - 43
NJETABLE parameter 1 - 43
NODEFAULT parameter 1 - 31
OPENJOBLIMIT parameter 1 - 47
PRIORITY parameter 1 - 34
QUEUEDEFAULT parameter 1 - 33
RESPONSEFILTER parameter 1 - 21, 1 - 24
SESSIONS parameter 1 - 21, 1 - 34
SSI-specific parameters, SSISELECT 1 - 45
TRACEWORD parameter 1 - 22, 1 - 26, 1 - 34
TRANSLATE parameter 1 - 22, 1 - 35
UCSTABLE parameter 1 - 44, 1 - 45
VPRT-specific parameters,
 LOGICALUNITTABLE 1 - 46

SET JOB command 1 - 48
 DISPOSITION parameter 1 - 48
 job defaults parameters
 CHARx 1 - 49
 CLASS 1 - 49
 DESTINATION 1 - 50
 FCB 1 - 50
 FLASH 1 - 50
 FORMDEF 1 - 50
 FORMS 1 - 50
 HOLD 1 - 50
 INDENT 1 - 50
 LINECOUNT 1 - 50
 NOCHARx 1 - 49
 NOCLASS 1 - 49
 NODESTINATION 1 - 50
 NOFCB 1 - 50
 NOFLASH 1 - 50
 NOFORMDEF 1 - 50
 NOFORMS 1 - 50
 NOHOLD 1 - 50
 NOINDENT 1 - 50
 NOLINECOUNT 1 - 50
 NOPAGEDEF 1 - 51
 NOPRMODE 1 - 51
 NOTRC 1 - 51
 NOUCS 1 - 51
 NOWIDTH 1 - 51
 NOWRITER 1 - 51
 PAGEDEF 1 - 51
 PRMODE 1 - 51
 TRC 1 - 51
 UCS 1 - 51
 WIDTH 1 - 51
 WRITER 1 - 51
 PRIORITY parameter 1 - 49
 QUEUE parameter 1 - 49
SET LOG command 1 - 52
 CLASS parameter 1 - 52
 DESTINATION parameter 1 - 52
 FORMS parameter 1 - 53
 LOGLEVEL parameter 1 - 53
 MESSAGELEVEL parameter 1 - 54
 NODESTINATION parameter 1 - 52
 NOFORMS parameter 1 - 53
 NOSPINSYNCHRONIZE parameter 1 - 55
 NOUPPERCASE parameter 1 - 22, 1 - 55

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- SPINRECORDS parameter [1 - 54](#)
- SPINSYNCHRONIZE parameter [1 - 55](#)
- SPINTIME parameter [1 - 55](#)
- UPPERCASE parameter [1 - 22](#), [1 - 55](#)
- SET OUTPROCESSOR command [1 - 56](#)
 - BSD-specific parameters, SYSID [1 - 60](#)
 - CONVERTSELECTOR parameter [1 - 56](#)
 - DEFAULT parameter [1 - 57](#)
 - LOGLEVEL parameter [1 - 58](#)
 - MESSAGELEVEL parameter [1 - 58](#)
 - NJE-specific parameters, NJEINPROCESSOR [1 - 61](#)
 - NODEFAULT parameter [1 - 57](#)
 - SESSIONS parameter [1 - 59](#)
 - TRACEWORD parameter [1 - 59](#)
- SET QUEUE command [1 - 62](#)
 - BSD-specific parameters [1 - 70](#)
 - BANNER [1 - 70](#)
 - CONTROLFILE [1 - 70](#)
 - CONTROLSTRINGPREFIX [1 - 71](#), [1 - 78](#)
 - CONTROLSTRINGSUFFIX [1 - 72](#), [1 - 78](#)
 - FILTER [1 - 73](#)
 - LOCALHOSTNAME [1 - 74](#)
 - LOCALPORTS [1 - 75](#)
 - NOBANNER [1 - 70](#)
 - NOCONTROLSTRINGPREFIX [1 - 71](#), [1 - 78](#)
 - NOCONTROLSTRINGSUFFIX [1 - 72](#), [1 - 78](#)
 - REMOTEHOST [1 - 76](#), [1 - 79](#)
 - REMOTEPORT [1 - 76](#), [1 - 79](#)
 - REMOTEPRINTER [1 - 77](#)
 - CONTROLLINEEXIT parameter [1 - 71](#)
 - DEFAULT parameter [1 - 63](#)
 - DESCRIPTION parameter [1 - 63](#)
 - DISPATCHEXIT parameter [1 - 63](#)
 - DISPOSITION parameter [1 - 64](#)
 - FILENAMETIMESTAMP parameter [1 - 73](#)
 - INDENT parameter [1 - 73](#)
 - LINESIZE parameter [1 - 64](#)
 - LOCALDELAYPORTTIMER parameter [1 - 74](#)
 - LOGLEVEL parameter [1 - 65](#)
 - MESSAGELEVEL parameter [1 - 66](#)
 - NODEFAULT parameter [1 - 63](#)
 - NODISPOSITION parameter [1 - 64](#)
 - NOINDENT parameter [1 - 73](#)
 - NOWIDTH parameter [1 - 69](#)
 - OUTPROCESSOR parameter [1 - 66](#)
 - PAGESIZE parameter [1 - 67](#)
 - RETRYCOUNT parameter [1 - 67](#)
 - RETRYINTERVAL parameter [1 - 67](#)
 - SENDALLCOPIES parameter [1 - 68](#)
 - SESSIONS parameter [1 - 68](#)
 - TRANSLATE parameter [1 - 69](#)
 - WIDTH parameter [1 - 69](#)
- SET SPOOL command [1 - 81](#)
 - ALLOCATION parameter [1 - 81](#)
 - AVERAGERECORDS parameter [1 - 82](#)
 - BLOCKSIZE parameter [1 - 82](#)
 - DATACLASS parameter [1 - 82](#)
 - HIGHLEVELQUALIFIER parameter [1 - 83](#)
 - MANAGEMENTCLASS parameter [1 - 83](#)
 - PRIMARYSPACE parameter [1 - 83](#)
 - READBUFNO parameter [1 - 83](#)
 - SECONDARYSPACE parameter [1 - 84](#)
 - STORAGECLASS parameter [1 - 84](#)
 - UNIT parameter [1 - 84](#)
 - WRITEBUFNO parameter [1 - 84](#)
- SETTINGS parameter
 - SHOW ALTERNATECONSOLE command [1 - 86](#), [1 - 88](#)
 - SHOW EPS command [1 - 90](#)
 - SHOW INPROCESSOR command [1 - 93](#)
 - SHOW JOB command [1 - 96](#)
 - SHOW LOG command [1 - 99](#)
 - SHOW OUTPROCESSOR command [1 - 102](#)
 - SHOW QUEUE command [1 - 103](#)
 - SHOW SPOOL command [1 - 106](#)
- SHOW ALTERNATECONSOLE command [1 - 86](#), [1 - 88](#)
 - ALL parameter [1 - 86](#), [1 - 88](#)
 - CONSOLE parameter [1 - 86](#), [1 - 88](#)
 - EVERY parameter [1 - 86](#), [1 - 88](#)
 - LOG parameter [1 - 86](#), [1 - 88](#)
 - SETTINGS parameter [1 - 86](#), [1 - 88](#)
 - STATUS parameter [1 - 86](#), [1 - 88](#)
- SHOW commands, to display status [2 - 9](#)
- SHOW CONVERTER command [1 - 89](#)
- SHOW EPS command [1 - 90](#)
 - ALL parameter [1 - 90](#)
 - CONSOLE parameter [1 - 90](#)
 - COUNTERS parameter [1 - 90](#)
 - EVERY parameter [1 - 90](#)
 - INPROCESSOR parameter [1 - 90](#)
 - LOG parameter [1 - 90](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- MODEL parameter [1 - 90](#)
- OBJECTS parameter [1 - 90](#)
- OUTPROCESSOR parameter [1 - 91](#)
- QUEUES parameter [1 - 91](#)
- SETTINGS parameter [1 - 90](#)
- STATUS parameter [1 - 90](#)
- SHOW INPROCESSOR command [1 - 93](#)
 - ALL parameter [1 - 93](#)
 - CONSOLE parameter [1 - 93](#)
 - COUNTERS parameter [1 - 93](#)
 - EVERY parameter [1 - 93](#)
 - LOG parameter [1 - 93](#)
 - MODEL parameter [1 - 93](#)
 - SETTINGS parameter [1 - 93](#)
 - STATUS parameter [1 - 93](#)
- SHOW JOB command [1 - 96](#)
 - ALL parameter [1 - 96](#)
 - ATTRIBUTES [1 - 96](#)
 - CONSOLE parameter [1 - 96](#)
 - COUNTERS parameter [1 - 96](#)
 - EVERY parameter [1 - 96](#)
 - LOG parameter [1 - 96](#)
 - sample with EVERY and ALL parameters [1 - 97](#)
 - SETTINGS parameter [1 - 96](#)
 - STATUS parameter [1 - 97](#)
- SHOW LOG command [1 - 99](#)
 - ALL parameter [1 - 99](#)
 - CONSOLE parameter [1 - 99](#)
 - COUNTERS parameter [1 - 99](#)
 - EVERY parameter [1 - 99](#)
 - LOG parameter [1 - 99](#)
 - sample with EVERY and ALL parameters [1 - 99](#)
 - SETTINGS parameter [1 - 99](#)
 - STATUS parameter [1 - 99](#)
- SHOW OUTPROCESSOR command [1 - 101](#)
 - ALL parameter [1 - 101](#)
 - CONSOLE parameter [1 - 101](#)
 - COUNTERS parameter [1 - 101](#)
 - EVERY parameter [1 - 101](#)
 - JOBS parameter [1 - 101](#)
 - LOG parameter [1 - 101](#)
 - MODEL parameter [1 - 101](#)
 - NOCONSOLE parameter [1 - 101](#)
 - NOLOG parameter [1 - 101](#)
 - QUEUES parameter [1 - 101](#)
 - sample with EVERY and ALL parameters [1 - 102](#)
- SETTINGS parameter [1 - 102](#)
- STATUS parameter [1 - 102](#)
- SHOW QUEUE command [1 - 103](#)
 - ALL parameter [1 - 103](#)
 - CONSOLE parameter [1 - 103](#)
 - COUNTERS parameter [1 - 103](#)
 - EVERY parameter [1 - 103](#)
 - JOBS parameter [1 - 103](#)
 - LOG parameter [1 - 103](#)
 - MODEL parameter [1 - 103](#)
 - NOCONSOLE parameter [1 - 103](#)
 - NOLOG parameter [1 - 103](#)
 - sample with EVERY and ALL parameters [1 - 103](#)
 - SETTINGS parameter [1 - 103](#)
 - STATUS parameter [1 - 103](#)
- SHOW SPOOL command [1 - 106](#)
 - ALL parameter [1 - 106](#)
 - ALLJOBS parameter [1 - 106](#)
 - CONSOLE parameter [1 - 106](#)
 - COUNTERS parameter [1 - 106](#)
 - EVERY parameter [1 - 106](#)
 - JOBS parameter [1 - 106](#)
 - LOG parameter [1 - 106](#)
 - NOCONSOLE parameter [1 - 106](#)
 - NOLOG parameter [1 - 106](#)
 - ORPHANS parameter [1 - 106](#)
 - sample with EVERY and ALL parameters [1 - 107](#)
 - SETTINGS parameter [1 - 106](#)
 - STATUS parameter [1 - 106](#)
- SHUTDOWN command [1 - 109](#)
 - DRAIN parameter [1 - 109](#)
 - FORCE parameter [1 - 109](#)
 - IMMEDIATE parameter [1 - 109](#)
 - SOLVE:EPS-Admin [C - 4](#)
- Shutdown SOLVE:EPS, SOLVE:EPS_Admin,
SOLVE:EPS [C - 12](#)
- SOLVE:EPS
 - executing as a batch job [2 - 8](#)
 - executing as a started task [2 - 8](#)
 - operator command usage [2 - 9](#)
 - recovery after unexpected termination [2 - 8](#)
 - SOLVE:EPS_Admin [C - 12](#)
 - starting [2 - 8](#)
 - stopping [2 - 8](#)
 - tracing facility [2 - 9](#)
- SOLVE:EPS-Admin [C - 11](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- commands [C - 4](#)
 - EXECUTE [C - 4](#)
 - NEWQUEUE [C - 4](#)
 - OPTIONS [C - 4](#)
 - REFRESH [C - 4](#)
 - RELOAD [C - 4](#)
 - SHUTDOWN [C - 4](#)
 - SPIN [C - 5](#)
 - Inprocessor [C - 11](#)
 - Jobs [C - 10](#)
 - Job Filter [C - 10](#)
 - Reset Job [C - 10](#)
 - Zoom Job [C - 10](#)
 - Logs [C - 11](#)
 - Logs, View/Change SOLVE:EPS Settings [C - 11](#)
 - Outprocessors
 - Dummy [C - 8](#)
 - Zoom [C - 8](#)
 - overview [C - 3](#)
 - parameters [C - 6](#)
 - Debug [C - 7](#)
 - Inactivity Timeout [C - 6](#)
 - Native Interface Modeling [C - 7](#)
 - Remote Machine Name [C - 6](#)
 - Remote Password [C - 6](#)
 - Remote Port [C - 6](#)
 - Remote User ID [C - 6](#)
 - TCP Address Space Name [C - 6](#)
 - queues
 - New Queue [C - 9](#)
 - Queue Filter [C - 8](#)
 - Route Queue [C - 9](#)
 - Zoom [C - 9](#)
 - Queues and Outprocessors [C - 8](#)
 - SOLVE:EPS [C - 12](#)
 - Reload Table [C - 12](#)
 - Shutdown SOLVE:EPS [C - 12](#)
 - View SOLVE:EPS Counters [C - 12](#)
 - View/Change SOLVE:EPS Settings [C - 12](#)
 - Spool [C - 11](#)
 - View Counters: Job Dispatch [C - 11](#)
 - View/Change Spool Settings [C - 11](#)
 - ViewCounters: Attempted Job Creation [C - 11](#)
 - SOLVE:EPS-Administration, See SOLVE:EPS-Admin
 - SOSI DBCS handling options [3 - 2](#), [3 - 5](#)
 - SPIN command [1 - 109](#)
 - SOLVE:EPS-Admin [C - 5](#)
 - SPINRECORDS parameter, SET LOG command [1 - 54](#)
 - SPINSYNCHRONIZE parameter, SET LOG command [1 - 55](#)
 - SPINTIME parameter, SET LOG command [1 - 55](#)
 - Spool, SOLVE:EPS-Admin [C - 11](#)
 - SSISELECT parameter
 - REFRESH command [1 - 14](#)
 - SET INPROCESSOR command [1 - 45](#)
 - SSI-specific parameters, SET INPROCESSOR command, SSISELECT [1 - 45](#)
 - START command [1 - 110](#)
 - LOGICALUNITNAME parameter [1 - 110](#), [1 - 111](#)
 - NJE parameter [1 - 110](#), [1 - 111](#)
 - starting SOLVE:EPS [2 - 8](#)
 - See START command.
 - statement types, User Exit Language
 - assignment [B - 8](#)
 - break [B - 9](#)
 - comments [B - 9](#)
 - conditional [B - 6](#)
 - STATUS parameter
 - PURGE command [1 - 13](#)
 - RELEASE command [1 - 15](#)
 - SHOW ALTERNATECONSOLE command [1 - 86](#), [1 - 88](#)
 - SHOW EPS command [1 - 90](#)
 - SHOW INPROCESSOR command [1 - 93](#)
 - SHOW JOB command [1 - 97](#)
 - SHOW LOG command [1 - 99](#)
 - SHOW OUTPROCESSOR command [1 - 102](#)
 - SHOW QUEUE command [1 - 103](#)
 - SHOW SPOOL command [1 - 106](#)
 - status, display with SHOW commands [2 - 9](#)
 - STOP command [1 - 111](#)
 - DRAIN parameter [1 - 111](#)
 - FORCE parameter [1 - 111](#)
 - IMMEDIATE parameter [1 - 111](#)
 - SESSION parameter [1 - 111](#)
 - stopping SOLVE:EPS [2 - 8](#)
 - See SHUTDOWN command.
 - STORAGECLASS parameter, SET SPOOL command [1 - 84](#)
-

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

STORCLAS, alias for STORAGECLASS parameter
of SET SPOOL command [1 - 84](#)

string literals, User Exit Language data category [B - 3](#)

strings, User Exit Language data type [B - 2](#)

subtype field, querying in Dataset User Exit [A - 30](#)

SYSID parameter

SET INPROCESSOR command [1 - 21](#), [1 - 26](#), [1 - 40](#)

SET OUTPROCESSOR command [1 - 60](#)

T

T07SCVCH

execution options [2 - 7](#)

JCL description [2 - 7](#)

sample JCL [2 - 5](#)

tables

ascii object fields

JES-to-ASCII converter exit [A - 46](#)

LU1 converter exit [A - 38](#)

Binary converter exit object fields [A - 42](#)

converter object fields [A - 54](#)

CONVERTSELECTOR exit

converter object name field [A - 33](#)

object fields [A - 32](#)

rc object fields [A - 33](#)

return object fields [A - 33](#)

Dataset User Exit

boolean object fields [A - 24](#)

call object fields [A - 22](#)

calltype object fields [A - 24](#)

data object fields [A - 20](#)

dataset object fields [A - 21](#)

datasubtype object fields [A - 27](#)

datatype object fields [A - 26](#)

job object fields [A - 22](#)

jobdisposition object fields [A - 25](#)

queue object fields [A - 21](#)

rc object fields [A - 24](#)

return object fields [A - 18](#), [A - 25](#)

returnstring object fields [A - 24](#)

stringoptions object fields [A - 26](#)

time object fields [A - 25](#)

time structure object fields [A - 19](#)

dbcssupport object fields [A - 56](#)

Dispatch User Exit

boolean object fields [A - 10](#)

data object fields [A - 7](#)

job object fields [A - 9](#)

object fields [A - 12](#)

queue object fields [A - 8](#)

rc object fields [A - 11](#)

return object fields [A - 11](#)

ebcdic object fields

JES-to-ASCII converter exit [A - 46](#)

LU1 converter exit [A - 37](#)

formfeed object fields [A - 57](#)

horizontal format object fields for LU1 converter
exit [A - 38](#)

JES-to-ASCII converter exit

object fields [A - 44](#)

options object fields [A - 45](#)

language object fields [A - 54](#)

LU1 converter exit

object fields [A - 35](#)

option fields [A - 36](#)

scscharacter object fields, LU1 converter exit [A - 39](#)

scstype object fields for LU1 converter exit [A - 40](#)

Translate/Only converter exit object fields [A - 42](#)

translatedirection object fields [A - 56](#)

vertical format object fields for LU1 converter exit
[A - 39](#)

TCP Address Space Name, SOLVE:EPS-Admin
parameter [C - 6](#)

THRU parameter

HOLD command [1 - 11](#)

PURGE command [1 - 13](#)

RELEASE command [1 - 15](#)

trace messages, types of [2 - 13](#)

TRACEWORD parameter

SET INPROCESSOR command [1 - 22](#), [1 - 26](#), [1 - 34](#)

SET OUTPROCESSOR command [1 - 59](#)

TRACEWORD parameter, SET EPS command [1 - 30](#)

TRANSLATE parameter

REFRESH command [1 - 14](#)

SET INPROCESSOR command [1 - 22](#), [1 - 35](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

SET QUEUE command [1 - 69](#)
Translate/Only converter exit [A - 40](#)
 object fields table [A - 42](#)
 rc object fields table [A - 55](#)
 return object fields table [A - 55](#)
translatedirection object fields table [A - 56](#)
TRC parameter
 SET INPROCESSOR command [1 - 38](#)
 SET JOB command [1 - 51](#)
TYPE parameter
 DEFINE ALTERNATECONSOLE command [1 - 6](#)
 DEFINE INPROCESSOR command [1 - 7](#)
 DEFINE OUTPROCESSOR command [1 - 8](#)

U

UCS field, interpreting [2 - 4](#)
UCS parameter
 REFRESH command [1 - 14](#)
 SET INPROCESSOR command [1 - 38](#)
 SET JOB command [1 - 51](#)
UCSTABLE parameter, SET INPROCESSOR
 command [1 - 44](#), [1 - 45](#)
UNIT parameter, SET SPOOL command [1 - 84](#)
UPPERCASE parameter
 SET EPS command [1 - 30](#)
 SET LOG command [1 - 22](#), [1 - 55](#)
User Exit 1, SOLVE:EPS [A - 3](#)
User Exit 2 [A - 4](#)
 parameter list offset [A - 4](#)
 return code values [A - 5](#)
User Exit Language
 data categories
 bit literals [B - 4](#)
 numeric literals [B - 3](#)
 objects with fields [B - 3](#)
 string literals [B - 3](#)
 data types
 bits [B - 2](#)
 numbers [B - 2](#)
 strings [B - 2](#)
Dataset User Exit [A - 15](#)
Dispatch User Exit [B - 2](#)

EPSUSEX [A - 6](#), [B - 2](#)
 keywords [B - 10](#)
statement types
 assignement [B - 8](#)
 break [B - 9](#)
 comments [B - 9](#)
 conditional [B - 6](#)
user exits
 Binary [A - 41](#)
 converter initialization [A - 34](#)
 CONVERTSELECTOR [A - 1](#), [A - 32](#)
 Dispatch [A - 6](#)
 JES-to-ASCII [A - 43](#)
 LU1 [A - 34](#)
 overview [A - 2](#)
 Translate/Only [A - 40](#)
USER parameter
 HOLD command [1 - 11](#)
 PURGE command [1 - 13](#)
 RELEASE command [1 - 15](#)

V

vertical format object fields table, LU1 converter exit
 [A - 39](#)
View Counters: Attempted Job Creation,
 SOLVE:EPS-Admin Spool [C - 11](#)
View Counters: Job Dispatch, SOLVE:EPS-Admin
 Spool [C - 11](#)
View Inprocessor Counters, SOLVE:EPS-Admin
 Zoom Inprocessor [C - 11](#)
View Job Attributes, SOLVE:EPS-Admin Zoom Job
 [C - 10](#)
View Job Counters, SOLVE:EPS-Admin Zoom Job
 [C - 10](#)
View SOLVE:EPS Counters, SOLVE:EPS_Admin,
 SOLVE:EPS [C - 12](#)
View/Change BSD Inprocessor Settings,
 SOLVE:EPS-Admin Zoom Inprocessor [C - 11](#)
View/Change Inprocessor Settings,
 SOLVE:EPS-Admin Zoom Inprocessor [C - 11](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

View/Change Job Settings, SOLVE:EPS-Admin
Zoom Job [C - 10](#)

View/Change NJE Inprocessor Settings,
SOLVE:EPS-Admin Zoom Inprocessor [C - 11](#)

View/Change SOLVE:EPS Setting
SOLVE:EPS_Admin, SOLVE:EPS [C - 12](#)

View/Change SOLVE:EPS Settings
SOLVE:EPS-Admin Logs [C - 11](#)

View/Change Spool Settings, SOLVE:EPS-Admin
Spool [C - 11](#)

View/Change SSI Inprocessor Settings,
SOLVE:EPS-Admin Zoom Inprocessor [C - 11](#)

View/Change VPRT Inprocessor Settings,
SOLVE:EPS-Admin Zoom Inprocessor [C - 11](#)

VPRT-specific parameters, SET INPROCESSOR
command [1 - 46](#)
LOGICALUNITTABLE [1 - 46](#)

W

WIDTH parameter
SET INPROCESSOR command [1 - 38](#)
SET JOB command [1 - 51](#)
SET QUEUE command [1 - 69](#)

WRITEBUFNO parameter, SET SPOOL command [1 - 84](#)

WRITER parameter
SET INPROCESSOR command [1 - 39](#)
SET JOB command [1 - 51](#)

Z

Zoom Inprocessor, SOLVE:EPS-Admin [C - 11](#)
Zoom Job, SOLVE:EPS-Admin Jobs [C - 10](#)
Zoom Outprocessor, SOLVE:EPS-Admin [C - 8](#)
Zoom Queue, SOLVE:EPS-Admin [C - 9](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z